Univerza *v Ljubljani*
Fakulteta *za gradbeništvo*
in geodezijo

JULIO JOAQUIN SALABERRI MARTIN

AUTOMATED BIM QA/QC PROCEDURES WITH PRACTICAL APPLICATION

AVTOMATIZIRANE BIM QA/QC PROCEDURE S PRAKTIČNO UPORABO

BIM A+ European Master in Building Information Modelling

Master thesis No.:

Supervisor:
Assist. Prof. Tomo Cerovšek, Ph.D.

Co-supervisor:
Andraž Starc
Franc Sinur, Ph.D.

Ljubljana, 2022

Co-funded by the Erasmus+ Programme of the European Union

ERRATA

Page	Line	Error	Correction

»This page is intentionally blank«

BIBLIOGRAFSKO – DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK: **004.728.8:004.946.5(043.3)**

Avtor: **Julio Joaquin Salaberri Martin**

Mentor: **doc. dr. Tomo Cerovšek**

Somentor: **Andraž Starc / dr. Franc Sinur**

Naslov: **Avtomatizirane BIM QA/QC procedure s praktično uporabo**

Tip dokumenta: **magistrsko delo**

Obseg in oprema: **80 str., 41 sl., 4 pregl., 2 pril.**

Ključne besede: **kakovost podatkov, QA/QC, preverjanje na osnovi pravil, avtomatizirano preverjanje modelov, IFC, Revit**

Izvleček:

Vse večja uporaba metodologije BIM na gradbenih projektih in večja zrelosti BIM v organizacijah ter projektnih skupinah void k večjim količinam in raznolikosti zahtev, ki jih morajo informacije v modelih BIM izpolnjevati, od gradbenih predpisov, do dopolnitev skozi faze prejekta, pa vse do upravljanja in vzdrževanja gradbenih objektov.

Projektantska podjetja morajo te zahteve upoštevati pri razvoju svojih načrtovalskih rešitev, odgovorna so za izdelavo geometrije in informacij v skladu s temi zahtevami. Za to morajo imeti projektatnska podjetja ustrezne načrte in postopke za zagotavljanje kakovosti in nadzor kakovosti, ki zagotavljajo ustreznost ustvarjenih projektnih informacij. Preverjanje modela je nujna dejavnost kot del načrtov QA/QC, da se zagotovi skladnost med izdelanimi informacijami in zahtevami. Vendar se v mnogih projektantskih podjetjih to preverjanje modelov še vedno izvaja z ročnimi ali polavtomatskimi postopki, ki zahtevajo občutno večjo količino časa in s tem tudi človeških virov.

Namen te magistrske naloge je opredeliti generični delovni tok za implementacijo avtomatiziranih preverjanj na podlagi pravil, ki omogočajo validacijo vsebine alfanumeričnih informacij razvitih modelov BIM skozi faze življenjskega cikla projekta v avtorskem okolju BIM. Pristop obravnavamo z vidika projektantskega podjetja in na osnovi temeljitega pregleda literature. Delovni tok se lahko izvaja izolirano ali pa se lahko integrira v procese skozi faze načrtovanja projektantskega podjetja. Uporabnost predvidenega delotoka je bilča preizkušena v študiji primera z uporabo orodij, ki so na voljo na trgu in omogočajo avtomatizirano preverjanje alfanumeričnih informacij modela ter ovrednotenje skladnosti teh informacij z informacijskimi zahtevami projekta.

»This page is intentionally blank«

BIBLIOGRAPHIC– DOKUMENTALISTIC INFORMATION AND ABSTRACT

UDC: **004.728.8:004.946.5(043.3)**

Author: **Julio Joaquin Salaberri Martin**

Supervisor: **Assist. Prof. Tomo Cerovšek, Ph.D.**

Co-supervisor: **Andraž Starc / Franc Sinur, Ph.D.**

Title: **Automated BIM QA/QC procedures with practical application**

Document type: **Master thesis**

Scope and tools: **80 p., 41 fig., 4 tab., 2 ann.**

Keywords: **Data Quality, Information Quality, QA/QC, Rule-based Checking, Automated Checking, IFC, Model Checking, Revit**

Abstract:

The increase of adoption of BIM methodology in construction projects, and the increase of BIM maturity in organizations and project teams, has led to an increase of the amount and diversity of requirements that the information in the BIM models need to satisfy, from building regulations, through the construction process, all the way to the built asset management.

Design companies need to take into account these requirements for the development of their design, and they are responsible of producing their design and information in compliance with those requirements. For this, design companies should have proper Quality Assurance and Quality Control plans and processes that ensure the quality of the information produced. Model checking is a necessary activity as part of the QA/QC plans to ensure that compliance between the information and requirements. However, still in many design companies this model checking is carried out by manual or semi-automated procedures that need a tangible amount of time and therefore, human resources.

This thesis aims to define a generic workflow to implement automated rule-based checks that allow to validate the alphanumerical information content of the developed BIM models in the design stages of a project life cycle in the environment of the BIM Authoring Tool, from the perspective of a design company, based on the information available in the literature. The workflow can be carried out in an isolated manner or can be integrated into the design phase processes of such design company. Afterwards, the applicability of the workflow has been tested in a case study, making use of tools that are available on the market and allow to perform automated checks on the alphanumerical information of the model and evaluate the compliance of this information with alphanumerical requirements.

»This page is intentionally blank«

ACKNOWLEDGEMENTS

To Tomo Cerovšek, for his patience and his guidance.

To Andraž Starc, for his support, optimism, and advice.

To IBE d.d., for providing me with material for the development of the thesis.

To my BIM A+ colleagues and friends, for their assistance during stressful times and joy during relaxing times; and to my friends at home, for always being available for me.

To the Consortium, for giving me the opportunity to join the program.

To my family and girlfriend, for their understanding and support in my decision to seek new knowledge and experience.

»This page is intentionally blank«

TABLE OF CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

LIST OF ABBREVIATIONS

AEC	Architecture, Engineering and Construction
AIM	Asset Information Model
AIR	Asset Information Requirements
BIM	Building Information Modelling
BMC	BIM-based model checking
EIR	Exchange Information Requirements
GUID	Globally Unique Identifier
ISO	International Organization for Standardization
LOD	Level of Development
OIR	Organizational Information Requirements
PIM	Project Information Model
PIR	Project Information Requirements
QA	Quality Assurance
QC	Quality Check
QM	Quality Management
RIBA	Royal Institute of British Architects
SMC	Solibri Model Checker

1 INTRODUCTION

With the passing of the years, the development of construction projects using BIM has increased, and in parallel, the maturity level of the project stakeholders in relation to BIM methodology, not only the AEC companies but also the clients, has also increased. The 3D models have passed from being just 3D graphical representations of the asset to be designed and built, to BIM models full of data that can be consulted and becomes information when put in context with the parametric objects that compose such model. This increase in the maturity level of the stakeholders in relation to BIM methodology has also translated into the necessity of the information available in the BIM models to be required to comply with more and more requirements of very different natures, from legal requirements for the design of a future built asset to operation and maintenance requirements for future assets, or even assets that already exist.

Design companies are one of the main actors that generate information in construction projects, especially when developing BIM models. In accordance with the ISO 19650 [1], all parties that generate information are responsible for such, and therefore having QA and QC plans and processes, that ensure that the information generated by the design company is of quality and the on the models is accurate, consistent and complete [2].

The use of BIM models boosts the quantity of information checked from 5-10% in traditional design processes up to 40-60% [3], and model checking is one of the activities that aims to check that information. But, still today, many design companies carry out this activity by manual or semi-automated procedures, spending human resources and time on it, which could be reduced if the model checking could be automated.

1.1 Objective and motivation

The present thesis aims to define a generic workflow that could be applied both independently and integrated into a design company´s QA/QC plans and processes during the design phase of a project, and that can allow to the company to implement rule-base automatic checks (automated procedures) that can be used to validate the alphanumerical information content in the BIM models and evaluate the compliance of this information with the alphanumerical information requirements for the project.

The previous work experience of the author of this thesis inside a design company and his familiarity with manual and semi-automated model checking procedures within the environment of the BIM Authoring Tool, have motivated the author to research the available literature, aiming to find processes that allow to implement automated procedures for model checking in the context of the design company and the Authoring Tool. The lack of references and literature on this matter have pushed the author to attempt to define a workflow to implement such automated model checking procedures. This interest

goes in alliance with the interest of the partner institution, IBE d.d., in finding and implement new procedures of model checking to their QA/QC plans that allow to increase the reliability of the information contained in BIM models by reducing human errors and, as a consequence, reduce the time that is needed to be spent in model checking, which could translate in costs saving for the company, and therefore, for the company´s clients.

1.2 Thesis Structure

The thesis has been structured in the following manner:

1. Introduction: this chapter aims to introduce the objective of the thesis and the motivations that led the author to develop his work on this topic.

2. Background: an intensive review of the available literature covering the topics of quality and its relationship with BIM, the proportional relationship between quality in the models and compliance with requirements, Quality Assurance (QA) and Quality Control (QC) in BIM and the difference between them, the concepts of validation, verification, code checking, model checking and the concept of automatic model checking and the information available on this topic.

3. Methodology: in this chapter, based on the information available in the literature, the author attempts to define a generic workflow that can be implemented into the design company processes and supports the use of automated model checking tools within the Authoring Tool environment during the design phase of the project, with the objective of validating the alphanumerical information content of the BIM models in relationship with the information requirements that is needed to be met.

4. Case Study: the workflow developed in the methodology will be tested into a BIM model provided by the partner institution, IBE d.d., with the intention of evaluating the applicability of the methodology defined by the author.

5. Conclusions

2 BACKGROUND

2.1 Quality in BIM

2.1.1 Introduction

Quality is a concept that can be applied to any field. For example, the Oxford Dictionary in its first definition, defines quality as "*the standard of something when it is compared to other things like it; how good or bad something is*" [4]. When finding sources that can define quality in relationship to BIM, the BIM Dictionary can be consulted. The BIM Dictionary is an Open Source & free to use project, developed by the BIMe Initiative. It defines quality as *"the measure of excellence or a state of being free from defects, deficiencies, and significant variations, brought about by the strict and consistent adherence to measurable and verifiable standards to achieve uniformity of output that satisfies specific customer or user requirements"* [5].

In ISO 19650, BIM is defined as "*use of a shared digital representation of a built asset to facilitate design, construction and operation processes to form a reliable basis for decisions"* [1]. In [6], Succar defines BIM as a set of technologies, processes and policies (standards) that result in a methodology to manage the building´s data throughout all its lifecycle.

The shared digital representation mentioned in the BIM definition given by ISO 19650 consists of digital assets, which can be 3D models, documents or data sets that represent the built asset [7]. These digital assets are, in the end, containers of information that is exchanged between all the stakeholders involved in the lifecycle.

Making a relationship between the previous concepts, it can be interpreted that quality in BIM involves, on the one hand, the quality of the digital assets, and on the other hand, the quality of the processes and standards. This goes in accordance with the expressed by Donato, Lo Turco and Bocconcino [6], who divide the quality evaluation of BIM processes into two: quality of the processes themselves and quality of the models. This thesis focuses on the quality of the models.

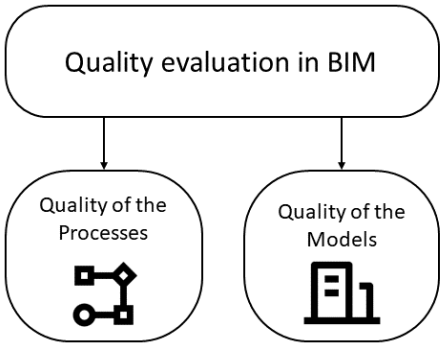

Figure 1: Quality evaluation in BIM (based on [8])

2.1.2 Quality of BIM models

In continuation with the expressed by Donato, Lo Turco and Bocconcino [8], the use of BIM methodology should translate into the generation of digital models containing unique and non-misleading information. The quality of the models is determined by the quality of the models entirely (is the model located in the correct place, do elements clash, …) and the objects contained in the model (do the elements contain the correct information to represent the built asset).

In simple words, a BIM model is a container that is used for exchanging information that describes a physical environment. The information contained in the model can be geometric or alphanumeric [9]. Thinking about a clash in a model, for example, it can be interpreted as the representation of a conflict in the geometrical information contained in the model, caused by defective geometrical information of certain elements in the model. Therefore, it can be deduced that a BIM model will be of good quality when the information that is contained in the model, both geometric and alphanumeric, is free of defects and satisfies the specified requirements.

2.1.3 Information in BIM models: poor data quality leads to wrong information

It is common to use the terms data and information to describe the same things. However, it is worth mentioning that in order for data to become information there needs to be a processing of this data [10]. "Concrete" is just data, but when it is associated to a wall in a 3D model, it becomes information. Therefore, poor data quality as an input translates into wrong information as an output.

Figure 2: Data to information (based on [10])

The "fitness for use" concept, adopted widely in the literature, led Wang and Strong [11] to define the concept of data quality as "*data that are fit for use by data consumers*" and the concept of data quality dimension as "*a set of attributes that represent a single aspect or construct of data quality*". In their work, the authors develop a conceptual framework for data quality, identifying up to nineteen dimensions of data quality, distributed into four different categories: intrinsic data quality, contextual data quality, representational data quality and accessibility data quality.

In addition to Wang and Strong [11] addressing the data quality, other authors, like Naumann and Rolker [12] or Delone and McLean [13], evaluate information quality and different dimensions that affect it.

Table 1: A conceptual Framework of Data Quality [11]

DATA QUALITY				
Data Quality Category	Intrinsic	Contextual	Representational	Accessibility
Data Quality Dimensions	Believability Accuracy Objectivity Reputation	Value-added Relevancy Timeliness Completeness Appropriate amount of data	Interpretability Ease of understanding Representational consistency Concise representation	Accessibility Access Security

In [2], Mirarchi and Pavan identify the most used objective dimensions of data quality as the following ones:

- Accuracy: to which extent the data is reliable, correct, and certified.
- Consistency: are the semantic rules defined over a data set satisfied.
- Completeness: is the data describing a set of real-world objects included in a particular data collection and to which extent.
- Timeliness: to which extent the data is up to date to represent an actual situation.

Based on the three first dimensions mentioned (leaving timeliness out of the analysis), the authors evaluate data quality issues that can appear in BIM models, developed with the same authoring tool (Autodesk Revit) by different design teams that were provided with the same set of requirements to comply with. They found common data quality issues across the different models, classifying them according to the data quality dimensions considered:

- Accuracy issues:
    - Inconsistency between information fields
    - Precompiled information fields:
        - Information is not known
        - Information is known but not used
    - Other human related accuracy issues (i.e., typos)

- Consistency issues:
    - Misuse of the object categorization provided by the tool
    - Information introduced in the wrong information field

- Completeness issues:
    - Objects do not satisfy information requirements defined in the EIR
    - Information structure defined to satisfy the requirements and volume of information generated

This analysis made by Mirarchi and Pavan [2], comes in response to the lack of exploration on the quality of possible data quality issues in native models. This possibility is mentioned in [14], where Solihin, Eastman and Lee analyse IFC quality validation, understanding how the export and import phases in the process are a cause of quality issues.

Based in their works, and without considering any quality dimensions, it could be deduced that quality issues in BIM models can be caused by two types of factors:

1. Human factors: data quality issues can appear when the persons responsible for adding data to the models do not do it in a proper way (i.e., processes for adding data are not followed correctly, the author does not have enough knowledge…)

2. Technology factors: data quality issues derived from the tools used to create or exchange the models (i.e., missing data caused by the exporting of a model to IFC, processing of empty fields by the tools…)

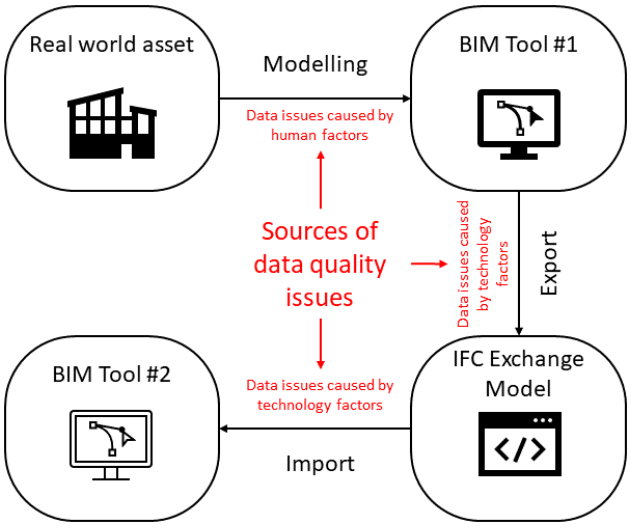

Figure 3: Sources of data quality issues in BIM models (inspired by [14])

2.1.4 Information Requirements

As mentioned earlier, quality depends on the satisfaction of the specified requirements. The ISO 19650 [1] defines information requirement as the "*specification for what, when, how and for whom information is to be produced*". The appointing party (usually known as the client) should understand and specify which information is needed to be delivered by the different lead appointed parties and appointed parties (i.e., designers, contractors…), that will be stored in the Asset Information Model (operational phase) or in the Project Information Model (delivery phase). In addition, appointed parties can add their own information requirements. The purposes for requiring information deliverables should be stated by the party requiring them, and they can include [1]:

- Asset register
- Support for compliance and regulatory responsibilities
- Risk management
- Support for business questions:
    - Management capacity and utilization
    - Management of security and surveillance
    - Support for renovation
    - Predicted and actual impacts
    - Operations
    - Maintenance and repair
    - Replacement
    - Decommissioning and disposal

These information requirements should be expressed in terms of project stages when related to the delivery phase (development of PIM) or in terms of life-cycle trigger events when associated with the operational phase (development of AIM) of the asset.

The management of the information requirements is one of the core concepts of the ISO 19650. If information requirements are not clearly defined, the possibility of delivery teams not being able to provide correct and complete information increases, which may negatively affect the decision-making processes and the progression of the projects.

The ISO 19650 defines four key documents that should be set out by the appointing party and that will define their information requirements:

- Organizational information requirements (OIR) [1]: describe information requirements that are necessary to answer or inform high-level strategic objectives of the organization.

- Asset information requirements (AIR) [1]: these requirements should address the information needed in order to answer the OIR that are related to asset and needed to support its operation and maintenance.
- Project information requirements (PIR) [1]: they should explain which information is needed in relation to the design, construction and purpose of the future built asset.
- Exchange information requirements (EIR) [1]: are defined in relation to an appointment and should be identified whenever any of them happens.. They are developed in response to the PIR and AIR needs. They include technical requirements (i.e., software tools to be used), management requirements (i.e., workflows), and commercial (i.e., information delivery dates). Their goal is to request the truly required information at each specified time.

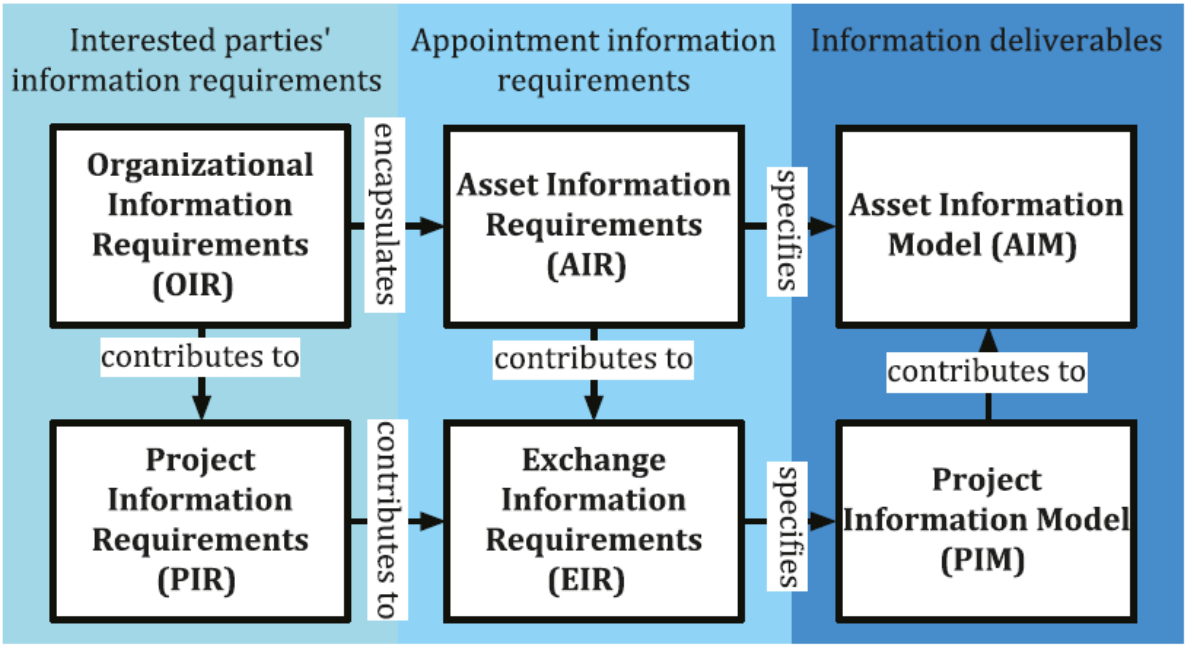

Figure 4: Hierarchy of information requirements [1]

2.1.5 From LOD to Level of Information Need

The concept of level of information need is introduced in the ISO 19650 as " *framework which defines the extent and granularity of information* " [1].

This concept arises in response to the concept of LOD (level of development) widely used in the industry with the intention of specifying the information that should be present in a BIM model. It was first introduced by the AIA (American Institute of Architects) [15], and after adapted by BimForum [16] to create the Level Of Development Specification (Part I and II) that is the most used LOD specification. However, what in the beginning was an attempt to standardize the information required in BIM models, the requirement of developing BIM models to a certain LOD can become a handicap due to evolution

of the use of BIM for projects of different characteristics and the application of BIM methodology by stakeholders of different backgrounds.

As an example, Mirarchi and Pavan [2] identify in their research that project teams that were served with the same LOD requirements for the development of BIM models, developed those models in totally different ways, one identified reason being the LOD specification not having being specifically developed for Italy, and other reason being the different cultural backgrounds of the project team members.

In order to adapt the concept of LOD to their market, some countries like Denmark [17] or Italy [18] have developed their own LOD specification. These initiatives bring into the market new specifications based on the same concept, using their own LOD levels (i.e., "LOD 300 DK" for Denmark or "LOD F" in Italy), which can bring confusion when stakeholders form different countries are involved in the same project.

All these LOD specifications relate the level of "geometrical" information in the model to a level of "alphanumerical" information, meaning that, if a certain level of geometrical information is desired, the alphanumerical information should also be developed up to a certain point. However, depending on the characteristics of the project, it may only be required to develop geometrical information, without any need of alphanumerical information (or the other way around). This can happen, for example, in renovation or decommission projects, where LOD might not be the right way to require information for the models.

The use of level of information need, instead, is determined according to the purpose of each deliverable, including the appropriate determination of granularity, quality and quantity of information [1]. The use of the level of information need to specify information requirements allows the party requiring them to be specific about the information that should be produced, acting with the end in mind and avoiding the production of information above the minimum amount needed to answer any requirement, which is considered as waste [1]. Different metrics to determine the level of information need should be defined and used across the whole project or asset, being all of this clearly described within the AIR, EIR, OIR or PIR [1].

The European Standard, EN 17412-1:2021: "Building Information Modelling - Level of Information Need - Part 1: Concepts and principles" [19] "*Establishes the concepts and principles for consistent detailing of the level of information need and information deliveries using building information modelling (BIM). Sets up a framework for specification, looking at the purpose, information delivery milestones, who is involved and the objects within a breakdown structure. Also considers the definitions, verification and validation*"[20] .

The purpose, milestones, actors involved, and the objects respond to the why (i.e., the information is needed to achieve a model use [21]), when (i.e., at the beginning of a RIBA Stage [22]), who should produce (i.e., structural designer) and what (i.e., structural columns) information is needed, while the level of information need answers how the information is needed. For the same purpose but a different object, the level of information need may vary [19].

For each object then the level of information need can be described as a set of [19]:

- Geometrical information:
    - Detail: describes the complexity of the object´s geometry in relation to the real object, ranging continuously from symbolic over simplified to detailed.
    - Dimensionality: the object will be characterized by spatial dimensions that can be 0D, 1D, 2D, 3D.
    - Location: describes the positioning and orientation of an object and can be absolute (in relation to a reference point) or relative (in relation to another object).
    - Appearance: describes the visual representation of the object compared to the real world, and ranges from symbolic to realistic.
    - Parametric behaviour: describes if the shape, positioning and orientation of the object are defined in a way that they are independent from other information associated to the object or to the context where the object is placed, allowing for partial or total reconfiguration. It can be fully requested, partially requested, or not requested at all.

- Alphanumerical information:
    - Identification: this is used in order to position an object within a breakdown structure. As an example, the identification of the object can be name, type name, classification, reference structuring, index, numbering…
    - Information content: it is the list of all the required properties, which can be grouped to facilitate the alphanumerical information management.

- Documentation: the documentation for an object or group of objects to support processes, decision making, approvals and information deliverables´ verification should be specified as a set of required documents.

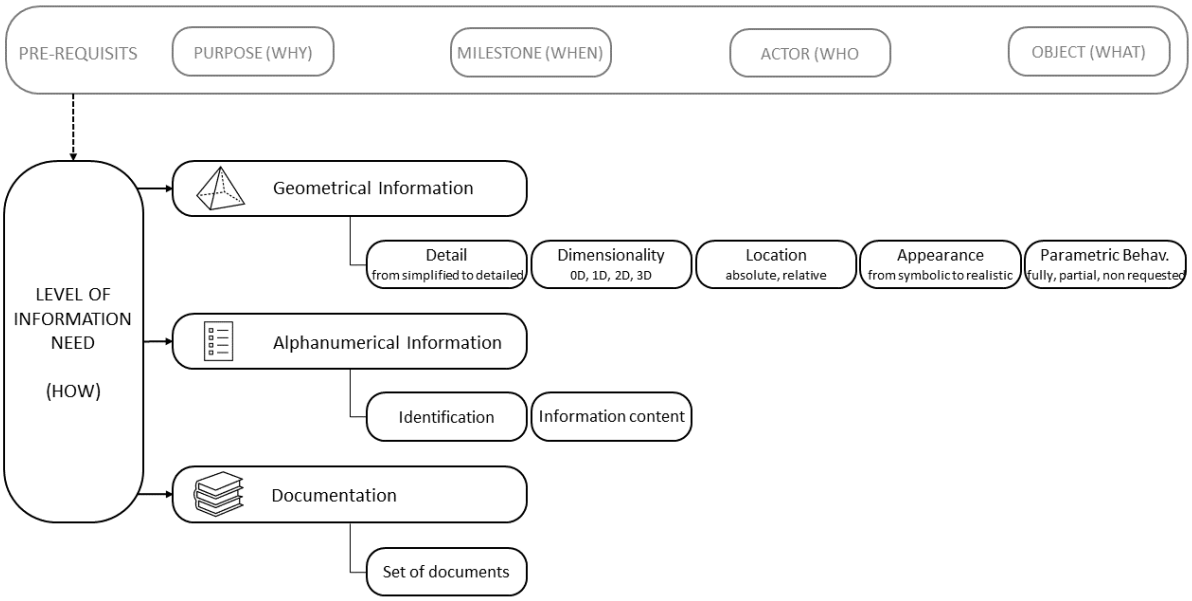

Figure 5: Level of information need relationships´ diagram (adapted from [20])

2.2 QA/QC and BIM models checking

2.2.1 Quality assurance and quality control definitions

On one hand, quality assurance (QA) can be defined as "*the system implemented to provide confidence that quality requirements for a product or service are met*" [23]. On the other hand, quality control (QC) can be defined as "*the activities conducted, or the mechanisms or techniques used to check if quality requirements for a product or service are met*" [24]. Both quality assurance and quality control are components of quality management (QM) along with quality planning and quality improvement [25].

Figure 6: Main components of quality management (based on [25])

It is frequent that both terms are used indistinctively to refer to the same thing calling it QA/QC process, and it is worth mentioning what is the difference between them. QA is a proactive practice designed to assure that a stated level of quality is reached, being its purpose to prevent defects from happening in the solution. QC, on the other hand, is a reactive practice to determine the quality of the product [26].

2.2.2 QA/QC in BIM models

To put into context the concepts of QA and QC in relation to BIM models, the following example can be used: modelling an object according to the company modelling standards is part of the QA plan defined for the model (it prevents the defect of an unproperly modelled object in the model), while checking if all objects in the model are modelled correctly according to the modelling standards is part of the QC. A QA plan for creating a BIM model up to a certain quality fulfilling the defined requirements can contain QC activities to ensure that the requirements that result on the model reaching the desired quality are met.

The Singapore BIM Guide Version 2.0 [27] mentions that it is the work of the BIM Manager to establish a proper QA plan for the models, that ensures appropriate checks on the information and the accuracy of the data, while the BIM coordinator of each discipline should be the one establishing a quality control procedure that ensures that each model is accurate and correct and it does not contradict the modelling guidelines. The guide also recommends considering the following aspects when defining a QA plan:

- Modelling guidelines
- Dataset validation
- Interference check
- Validation of BIM data to be used for Cross-Disciplinary Model Coordination

In addition, they provide a modelling guideline for QA for each discipline and different stages (Appendix C of the guide) [27].

Similarly to the Singapore BIM Guide [27], it is a common practice nowadays that public entities develop their own guides or manuals, each of them addressing the QA or QC processes according to their needs and depending on their maturity level. Using the case of Spain, public entities like Puertos del Estado [28], Euskal Trembide Sarrea [29], Junta de Extremadura [30] or Ferrocarrils de la Generalitat Valenciana [31] have published their own guides or manuals on BIM, covering the QA and QC topics in more or less depth. Therefore, it is worth mentioning that the vision on QA will be different if it is looked at it from the point of view of the client or the designer [32].

As mentioned by Donato, Lo Turco and Bocconcino [8], there are international standards and best practices on BIM implementation, while there is no clear one for quality assessment. In their work, they propose a semi-automatic procedure for the evaluation of the quality of the models and processes. In [33] Barichello proposes a framework for QA and its application in BIM authoring tools. As opposed to that, the topic on QA and QC is often found in literature being addressed by using IFC files [14] which is also encouraged, in practice, by organizations like buildingSmart [34]. In [35], Kovacs and Micsik take a step further away, and suggest eight principles for BIM-based compliance checking in which, if

they agree that quality control should be independent of the software used and should be interoperable and available as open source, being this also the principle behind the use of IFC, they also suggest that IFC is not enough flexible and there is still a lot of information commonly used in industry that is not specified in the scope of IFC.

In its conferences at BIMxp 2020, Vendrell [36], [37], defines quality management of models as a 2 stage process, QA taking place in the authoring environment, previously to the federation of models, while QC takes place in open environments with the use of IFC formats after the federation of models has taken place:

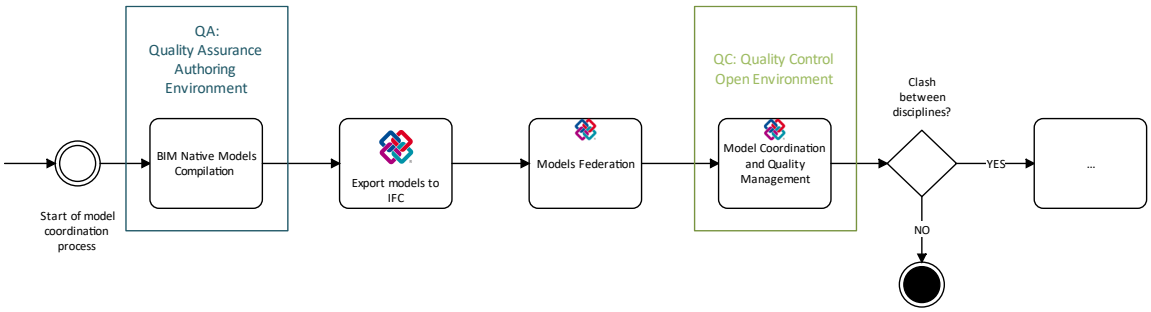

Figure 7: Quality management in BIM: a two stage process (adapted from [36], [37])

2.2.3 BIM Model Validation or Model Checking

Model validation or model checking (also referred to as model compliance checking, quality checking, model auditing…) is "*the process of checking models for possible data loss, data corruption or incompatibility with defined specifications*" [38]. Model checking is a powerful tool for QA and QC, but there should not be confusion between "quality checking" and "quality control" (QC).

The use of BIM models boosts the quantity of information checked from 5-10% in traditional design processes up to 40-60% [3]. Nowadays, there is an increasing interest in model checking tools that can run automatic checks as part of the QA and QC processes. Being able to check the informative content of the models and verify the presence of the alphanumeric information required in the EIR and BEP, can detect potential flaws in advance and the decision making relying on verifiable information contained in the model is guaranteed. Model checking is at the same time an essential an very powerful tool for the designer, as it allows to perform regular self-assessment [39]. The potential of implementing automated checks from the designer point of view would be able to increase the control on the information produced, being able to reduce future reworks and therefore saving time. The verification

and analysis of the results of model validations should be part of a standard routine, allocating proper time for it and for any adjustments on the model that may be required as a result of the analysis [3]. There are different ways of carrying out model checking, going from a simple visual check (i.e., identifying modelled elements that are not required or are not modelled in a proper way), to semi-automated or fully automated checks.

In [40], Hjelseth defines "BIM-based model checking" (BMC) as "*software which processes the content of information in BIM-files according to rules specified as pre-defined procedures*". For BMC he identifies the following components:

- Software: allows to read the BIM models, process rules, and present the results.
- Information in the BIM-file: the structure and the content of the information should comply with the requirements in the rules.
- Rule-sets: the collection of rules within a topic (i.e., clash detection). A rule is a logic question whose answer can be "yes", "no", or "not checked" if there is missing information. The BMC will read the information in the model as input for processing the rules.

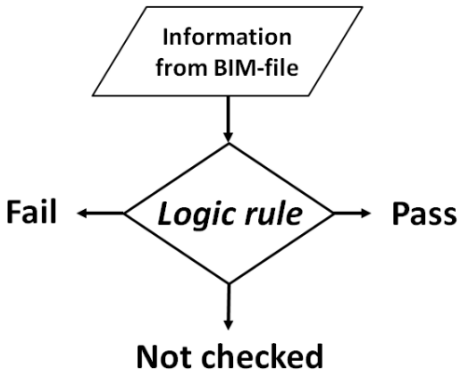

Figure 8: Principle of BMC [40]

In [41], Hjelseth proposes a framework to classify BIM checking concepts and defines the following classification on BMC:

1. Validation Checking [41]: Verifying that the BIM-designed solution adheres to the rule-sets` established guidelines is the goal of this concept. One or more sources, such as codes, standards, agreements, best practices, or other predetermined requirements, may serve as the foundation for a rule set. The most widely used sort of validation checking is clash detection, whereas code checking is a frequent type of checking but still with limited implementation.

2. Model content checking [41]: A defined set of pertinent data is compared to the BIM model content. Identified and not Identified are the two alternatives for this checking. Writing a report

to record compliance could be the next step. Specified data can also be removed, and default data can be used to replace properties.

3. Smart object checking [41]: the goal of smart object checking is to enable embedded pre-defined rules or algorithms to automatically adapt to the surroundings as the object itself observes it. The use of smart objects helps to update designs continuously. In the opinion of this thesis´ author, this concept does not fit the concept of checking or in any case it could be a subdivision of the first concept, validation checking, as stated by [35] as it is a real-time adaption of the object in relationship to the other objects in the 3D model.

4. Design option checking [41]: this check confronts the 3D model against a complex knowledge base resulting in different alternative design versions that fit the criteria. The solutions that allow this type of checks are still immature in the AEC industry, but one example of them could be "Generative Design for Revit" [42].

After defining the four major concepts types of BMC, Hjelseth [41] classifies them in two major concept groups: compliance checking and design solution checking:

Table 2: Overview of concepts of BIM-based model checking [41]

Concept group	Concept type	Purpose of checking	Outcome	Examples
Compliance checking	**Validation checking**	Validation	Pass/fail	Clash detections Code compliance
	Model content checking	Content of information	A filtered list	Relevant information for exchange
Design solution checking	**Smart object checking**	Integration (adaptation)	A modified model	Size of building parts (objects) related to the building (model)
	Design option checking	Guidance	Options and advice	Knowledge system for selecting relevant solutions

When the activity of checking the models is part of a QA or QC plan, the intent is not finding different design alternatives or define the quality of the design, meaning that the concept group of "design solution checking" are not included in such plans, but can be used as continuous support for the design process [41].

In practical application at project level, according to Cirbini, Mastrolembo and Bolpagni [39], the process of Model Validation is organized in three different phases of consequential verification (which are, in fact, checking types of the compliance checking concept group defined by Hjelseth [41]):

1. BIM Validation [39]: it essentially consists in the check of the geometrical and alphanumerical information in the model, checking also the if the procedures of modelling are correct. This phase takes place to ensure that the 3D model is prepared in a way to carry out further analysis based in the information contained in the model, the structure of the information, the naming and classification of the objects, etc. If the use of a rule-based model checking tool is expected, both the tool and the model to be checked should share the same semantics, therefore there will need to be a mapping in the checking tool to recognize the information in the 3D model. A successful validation of the model will verify that the data contained in it can be recognized by the checking software as the data is accurate, consistent, and complete, and that the model complies with the defined requirements. The verification and validation processes defined in the EN 17412:2 [19] should take place in this phase. The use of the level of information need framework allows the verification and the validation of the informative content in the BIM model, and it is potential to be machine-readable helps reducing the checking time and human errors. If the level of information need is unambiguously defined, it will avoid different interpretations of the same requirement, avoiding mistakes. The verification and validation of the information are defined as follows:

    - Verification [19]: "*confirmation, through the provision of objective evidence, that specified requirements have been fulfilled.*" The verification of deliverables against the level of information need can be used to check the existence of objects in the model (i.e., are there doors at a certain building story), alphanumerical information (i.e., do fire-protection doors contain the fire-rating property), geometrical information (i.e., does a door contain dimensions information) or documentation.

    - Validation [19]: "*confirmation, through the provision of objective evidence, that requirements for a specific intended use or application have been fulfilled*". It guarantees that the geometrical information, the alphanumerical information, and the documentation specified can be used for the defined purposes (i.e., validating that the value of the fire-rating property is a time type of value).

2. Clash Detection [39]: the process of clash detection is one of the most known model checking procedures when using BIM methodology, as it allows to perform a geometric and spatial analysis in a short time and identify current spatial interferences between the future build elements that would result in conflicts during the construction stage of the asset if not identified properly in advance.
   There are subsequent steps to clash detection: first, the designer should detect any geometrical interference to his competence their discipline model, like the architectural or the structural

model. After that, a clash detection between different disciplines is performed to detect any issues in the merged model happening and that may involve different designers [3].
Clash detection does not only serve the purpose of checking if two future built elements are physically affecting each other, as not having enough clear space around an element can also be a type of clash. There are, in essence, three types of clash detection [43]:

- Hard clash [43]: it's the most known of them, and the most basic, happening when two elements occupy the same specific space, for example, a duct passing through a column.

- Soft clash [43]: in this case two elements are not occupying the same physical space, but when one object does not have enough spatial tolerance around them. An example of a soft clash can be a column being placed in front of a door, close enough to it so a person cannot access through the mentioned door.

- Workflow clash [43]: also known as 4D clash, as opposed to hard clashes and soft clashes, these do not involve a physical space interference but involve conflicts in schedules, such as workflow timeline, material delivery, equipment delivery, etc. They are mostly scheduling related, and their existence translates into a decreasing of performance in the construction of the asset due to activities affection.

When performing a clash detection check, usually, the check is based on a "Clash Detection Matrix" that should be defined in the BEP of the project. This matrix is organized in a way that it gives information about which sets of elements will be checked against which, the type of the check, the tolerances needed, and the parties involved. Based on this matrix, the rule-sets for the clash detection check should be developed, and finally, the check should be performed and the results analysed.

3. Code Checking [39]; this type of check compares the design developed in a BIM model against codes and regulations [44]. Whereas the model validation and the clash detection check´s objective is just evaluating the informative content on the models without any judgement on the design of the solution adopted for the project, the code checking does check the design in order for it to be compliant with design codes, that are different depending on the location of the project. Due to this, those codes need to be translated into parametric rules that allow to be read and interpreted semantically, the same way that client requirements or other information that would be suitable to be checked should be also translated, so that after they can be implemented as rules [39]. The object elements need as well to be enriched with alphanumerical information that is not automatically created by the process of geometrical modelling, something that happens, for example, with dimensional data [45].

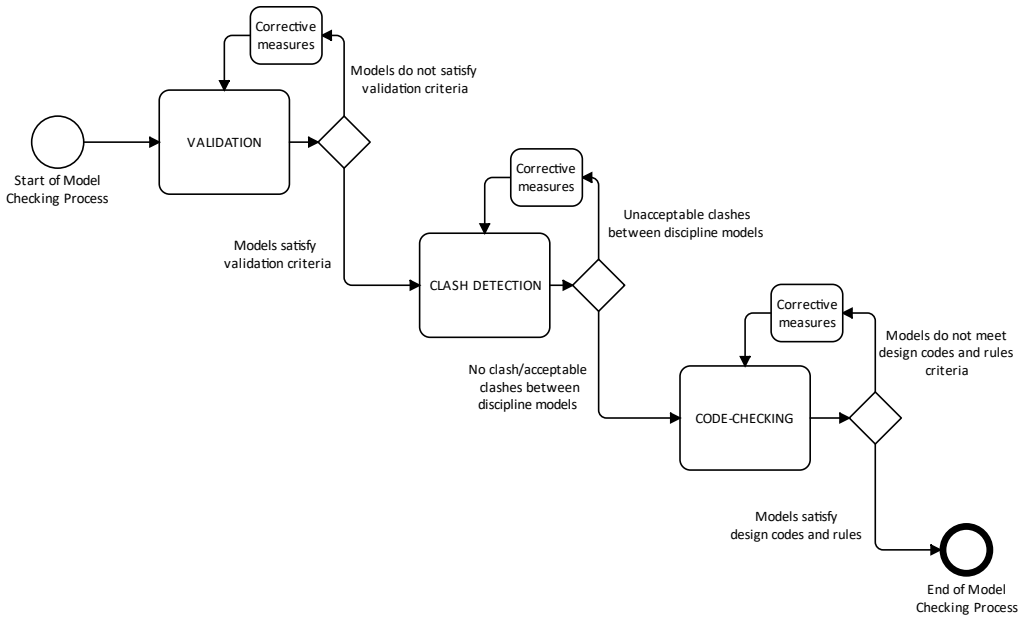

Figure 9: Consequential phases of model checking (inspired by [39])

2.2.4 Automated rule-based checking: four major stages

Eastman et al. [46] define automated rule checking as "*software that does not modify a building design, but rather assesses a design on the basis of the configuration of objects, their relations or attributes*", which is similar to the definition given by Hjelselth [40] to BIM-based model checking (BMC). In their work, they analyse different software platforms that support implementation of rule checking to be applied on IFC models, as they consider the IFC as the only neutral model representation that allows to describe a building for rule-checking. They also point out the potential emerging of rule-checking using native application data models.

Automated rule-based assessment tools can be implemented as: applications tied to a design tool, like plug-ins allowing to perform checks at the designer discretion, stand-alone applications running in parallel to the design tool or web-based applications accepting design from different sources [46].

In [46], the authors identify a necessary structure that allows to implement functionally complete rule-checking and reporting systems, and they structure the rule-checking into four major stages: Rule Interpretation, Building Model Preparation, Rule Execution and Rule Check Reporting.

1. Rule interpretation [46]: in this stage, the objective is to be able to translate and formalize the requirements needed, and that usually are expressed in a natural language, to machine-readable rules that allow to carry out the rule-based checking. This can be the most vital and complex

stage of the four stages defined. There are different approaches to carry out this translation, explored by Ismail, Ali and Iahad in [47]:

- Existing software or plug-in applications: this approach uses the capabilities of the software to perform this stage. The most famous software for this is Solibri Model Checker (SMC) [48].
- Object-based approach: this technique for organizing knowledge is done representing object types as the knowledge. Examples of organizing the knowledge are defining the attributes, rules, procedures, and machine learning. There are three stages to represent codes using this approach: building code classification and abstraction, rule representation modelling and knowledge base establishment. An example of this is CORENET ePlanCheck [49].
- Logical approach: the use of predicate logic is a common way to allow to map rules from natural language to a processable form. A predicate will be a well-defined term that can be translated into "true" or "false" and the rules can be connected through logical connections like "and", "or" and "if-then". Other methods for this are conceptual graph and decision table methodology.
- Ontological approach: an example of this is the RASE methodology (Requirement, Applicability, Selection and Exceptions) [50].

2. Building model preparation [46]: This stage involves the BIM model being built in a way that enables the quality checking. Therefore, its structure and the information richness should be suitable for performing the given rule-based check. For example, a stair modelled as a set of slabs could not be evaluated through rules that pretend to evaluate the parameters of stair objects.

3. Rule execution [46]: this stage performs the comparison of the information contained in the BIM model to the defined requirements and gives back the pass or fail of the rule-based check.

4. Rule check reporting [46]: this stage happens at the end when there is the need to communicate or evaluate the results of the checks. This reporting can go from as simple as a table with the results in each line to a 3D visualization with coloured objects that compose the BIM model. The report is basic to understand, communicate, follow, and solve whatever problems may be detected by the checks in an effective way.

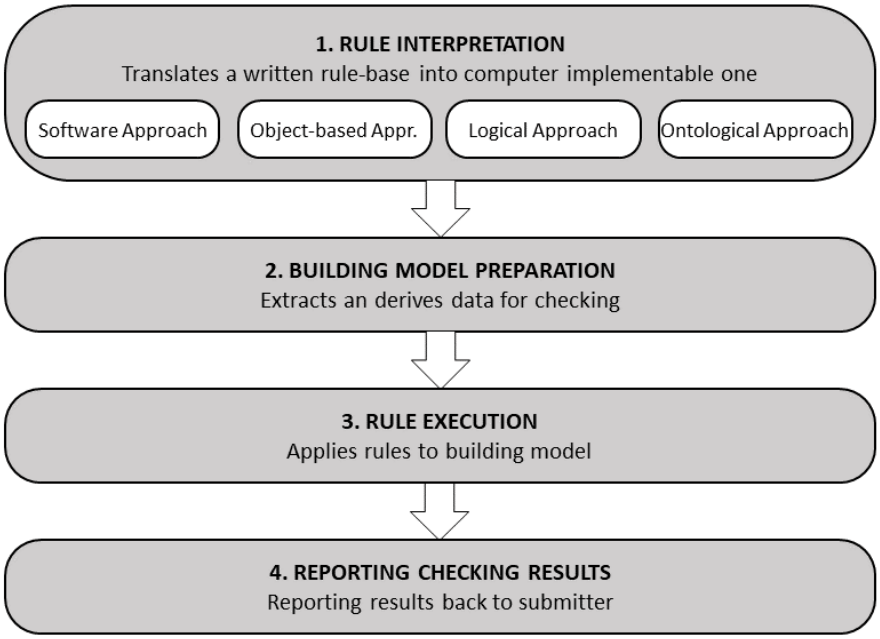

Figure 10: General structure of a rule-based checking process (inspired by [46], [47])

2.2.5 Classes of rules for automated rule-based checks

In [51], Solihin and Eastman carry out a classification on the types of rules for application in building models depending on the complexity of their computation and the requirements imposed on the rule execution environment. According to the authors, rules can be classified in 4 classes:

1. Class 1 – Rules that require a single or small number of explicit data [51]: the rules of this class check the information available in the object´s parameters, making it the most straightforward ones to implement, which, at the same time, grants them a huge potential on improving the model quality, especially when there is a consistent and easy way for the user to define a rule that checks basic information. For example, a rule that evaluates if the value of the parameter "Area" of an object is above a certain number.

2. Class 2 – Rules that require simple derived Attribute Values [51]: this class of checks base their rule on the combination of multiple parameters using custom algorithms that process data obtained from specific parameters to, afterwards, derive the information requested. For example, a rule that will combine the values of the parameter "Area" of different objects and then evaluate if the combined value is above a certain number.

3. Class 3 – Rules that require extended data structure [51]: this rules collects rules based on external data inputs. It is possible that "Class 1" and "Class 2" rules fall in this category if they

need external inputs. For example, a rule that will identify the "GUID" of an object and then evaluate the value of the "Area" that corresponds to that GUID in an excel table.

4. Class 4 – Rules that require a "proof of solution [51]: this collection of rules are not used with the objective of resulting in compliance or non-compliance, but require a "proof of solution". The application is more interested in the solution that complies with what the rules expect.

2.3 Summary

When referring to BIM methodology, the concept of quality can be related to the BIM processes or to the information models [8]. The BIM models are containers of data put into the context of the future built asset, becoming information, therefore the quality of the models is directly proportional to the quality of the data and its capability to comply with whatever requirements are expected to be satisfied [1].

The three principal dimensions of quality of data in BIM models are accuracy, consistency and completeness [2], and the appearance of issues in the quality of the data can be related to the user involved in the modelling process [2] or related to the translation of the data that may happen in interoperability processes (import/export of data) [14]. In addition, the production of more data than the minimum needed to satisfy the requirements can be considered as waste [1].

A Quality Assurance plan (QA), focusing on the authoring tool [36], aims to prevent these data issues from appearing in the model, and reactive measures to correct the issues that might be present in the models can be implemented with a proper Quality Control (QC) plan in open environment [37].

Model checking is a powerful tool for both, QA and QC, to validate the data contained in the BIM models [39], and the use of the "Level of Information Need" framework as a substitute of the typical LOD as a mean to require information contained in the BIM models improves the processes of verification and validation of the data contained in the models, clash detection and code checking [39] and, in addition, enables the potential of rule-based model checking procedures [19]. Rule-based model checking allows to perform fast and automatic compliance checks [40], [41], [46], [51], of that data, resulting in valuable information to ensure the quality of the models and support decision making related to the built asset.

3 METHODOLOGY

The present chapter of this thesis aims to define a "workflow" to develop automated rule-based checks to help validate the information produced in the BIM models during the design stages of the life cycle of a construction project that involves BIM methodology. Based on the four major stages of automated rule-checking defined by Eastman et al. [46], a workflow is proposed in order to develop automated checks that can be used as a tool of self-assessment and a way to control of the information produced by a design company that is part of the development of a project.

3.1 Problem statement and hypothesis

With the increase of adoption of BIM methodology in construction projects and the increase of the BIM maturity of the stakeholders involved in them, the information generated in the projects is also increasing in a directly proportional way. As signalled by the ISO 19650 [1], the task delivery teams of Appointed Parties are responsible for the information they generate and, therefore, for making sure that such information is accurate, reliable and usable by the Appointing Party as well as the other stakeholders involved in the collaborative production of information.

In general, in design companies, the checking of the information produced is still carried out by manual or traditional methods as part of the QA or QC processes, and it takes large amounts of time. The need of producing the information using an authoring tool, combined with the possibility that the same designer is involved in different disciplines of the project (which most likely translates into different disciplines making use of the same authoring tool to generate information) and the increased use of cloud platforms like Autodesk Construction Cloud [52] or Graphisoft BIMCloud [53], enhances the potential of checking the information directly in the authoring tool, avoiding the export to other formats like IFC to check the information produced and the possible information flaws derived from such export process.

Therefore, the implementation of automated checking procedures as part of the QA/QC processes in the authoring tool environment, can help save time that designers spend for that task, avoiding source of errors like exporting processes, and improve the efficiency of the information production and the control over such information, which, ultimately, translates into money saved by the client.

3.2 Overview

As mentioned, the aim of the "workflow" is to serve as support to implement automated rule-checks during the design stages of the project, to help improve the validation of the information produced in the BIM models, from the design company point of view. For the development of this workflow, it is assumed that the design company has already a certain maturity level and therefore they have their own defined processes and standards, and they apply BIM methodology as a daily basis.

3.2.1 Objectives and principles

The objectives of the implementation of the automated rule-based checks as a part of the design stages of the project life cycle are the following:

- Improve the validation process of the produced information through automated procedures.
- Ensure the accuracy, reliability and usability of the information contained in the BIM model.
- Increase the control over produced information during the design stages.
- Improve efficiency and reduced the amount of time necessary to carry out the checking processes.
- Serve as a tool for company self-assessment and supporting decision making during the design stages.
- Serve as a tool to support the modelling process.

In order to reach the defined objectives, the rule-based checking system should be able to support the following principles:

- Replicability: the developed rule-based checks should be suitable to be implemented across different projects with a reduced amount of effort in the adaptation of the rules from one project to another.
- Modularity: it should be possible to perform different checks not as a whole, but as separated checks that can be included or excluded depending, for example, on the current stage of the life cycle, or the set of information is pretended to be checked.
- Storage ability: the automated rule-based checking system should allow to storage different checks and support building a "check-set" library, that at the same time enable the replicability of the checks.

3.2.2 Workflow structure

The workflow defined in this thesis is based on the research performed in the "Background" chapter of this thesis, and specially based on the four major stages for Automatic rule-based checking defined by Eastman et al. [46]. The workflow structure can be seen in the following diagrams, and consists on the following steps:

1. Rule Interpretation.
2. Building Model Preparation.
3. Rule Execution.
4. Rule Checks Reporting.

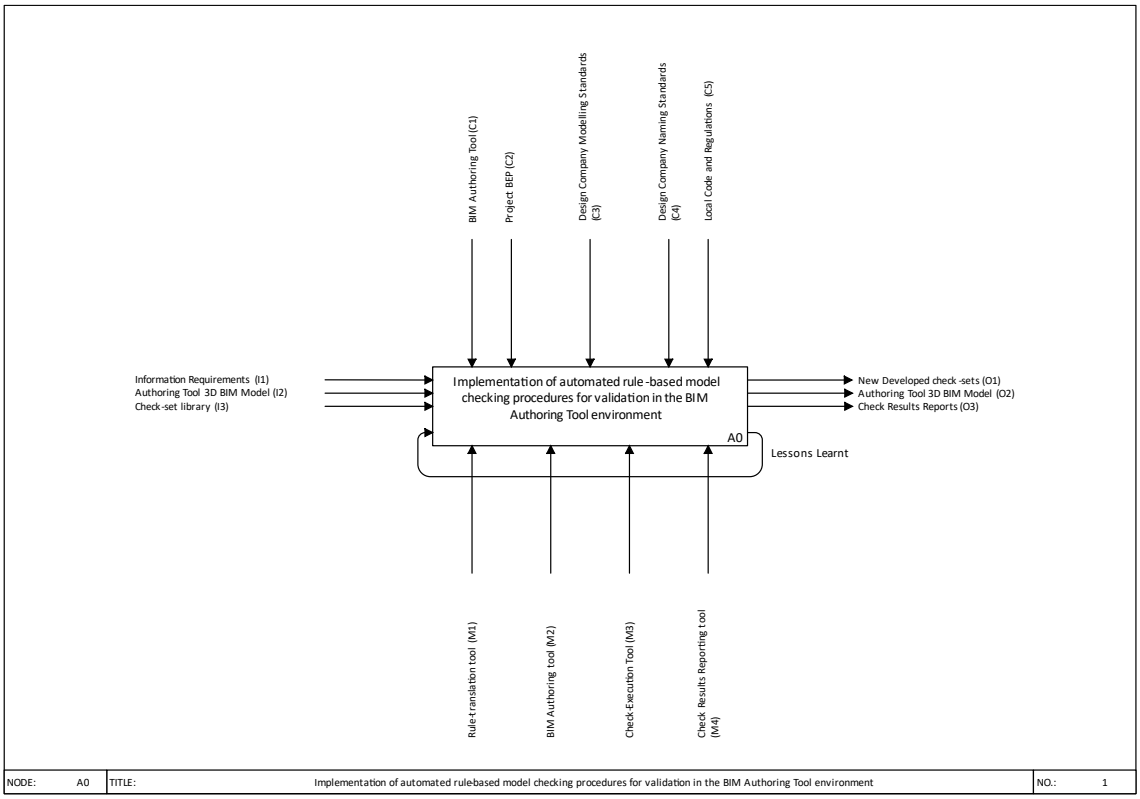

Figure 11: Context diagram for the workflow

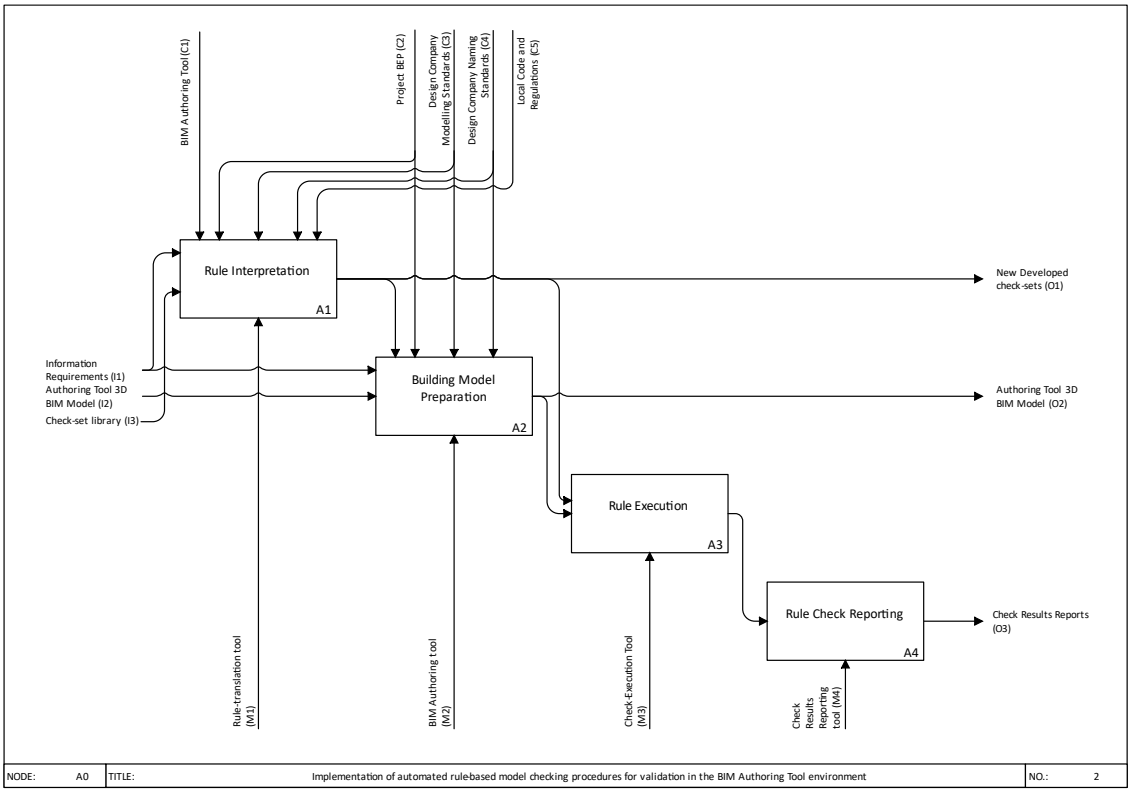

Figure 12: Diagram of the main steps of the workflow

3.3 Step 1: Rule Interpretation

The first step of the workflow is dedicated to make a translation from the different information requirements to machine readable rules that can be interpreted by the software tools that might be used.

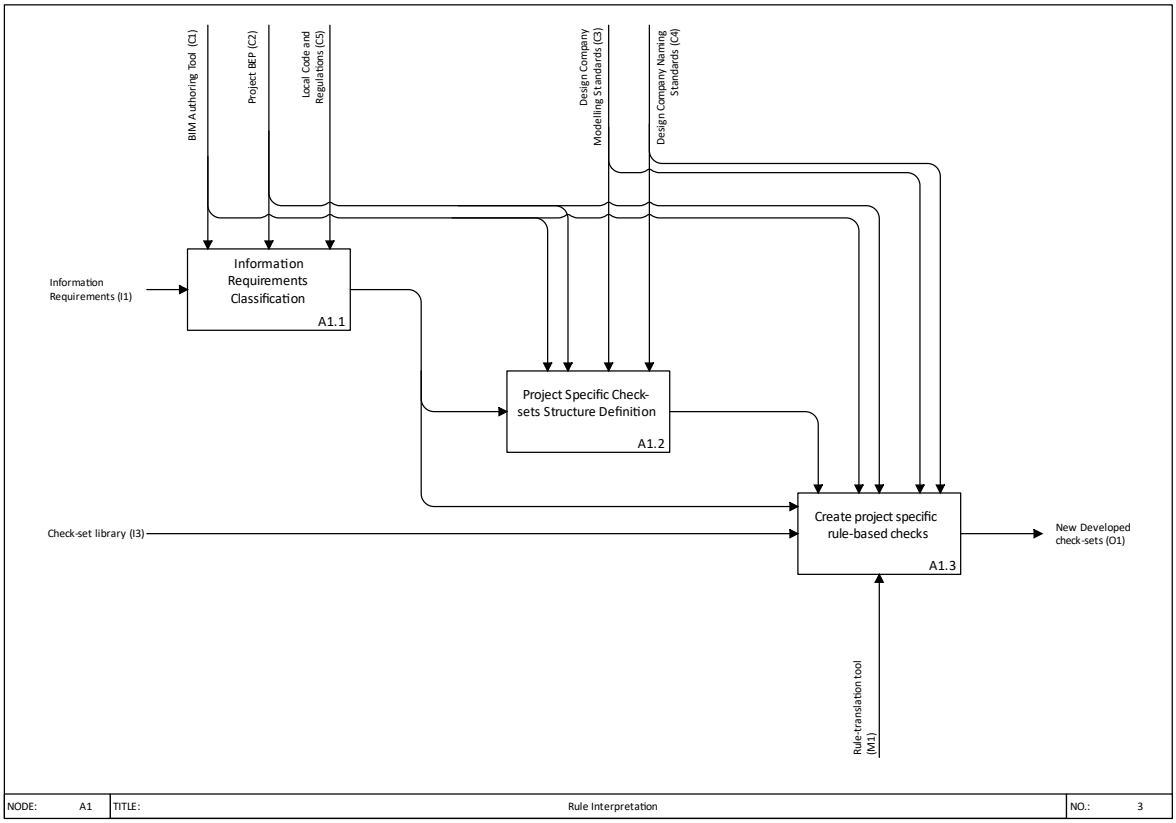

Figure 13: Diagram for Rule Interpretation step

3.3.1 Information Requirements Classification

The first activity for this step would be to classify the Information Requirements that the model needs to comply with. As a result, for this activity, there will be Information Requirements that can be translated into machine-readable rules that can evaluate the information content of the models or Information Requirements that cannot be translated into the machine-readable rules. The **input** for this activity is:

- Information Requirements (I1), which include: the OIR, AIR, EIR, PIR, and any information requirement that comes from other stakeholders of the project, as well as own information requirements from the design company.

For the case of Alphanumerical Information validation, it is ideal that the Information Requirements include the Level of Information Need. This Level of Information Need will specify the list of all

required properties for the objects contained in the BIM model, which enhances the implementation of automated rule-based checks.

The **controls** for this activity are:

- BIM Authoring Tool (C1): the way that the authoring tool structures the information conditions the way that information requirements are implemented in the BIM models, therefore, the Authoring Tool affects the classification of the Information Requirements.

- Project BEP (C2): the project´s BIM Execution Plan defines the strategy on how to address the Exchange Information Requirements. The BEP should include information such as delivery milestones, strategies, and other information relevant like for example, the QA plan. In addition, it is very important for this workflow that the BEP includes the Design Company´s Modelling Standards, Naming Standards, and the Project Model Breakdown Structure.

- Local codes and regulations (C5): these will vary depending on the location of the project.

The **output** for this activity will be the Information Requirements that can be classified as suitable to be "rule-implemented".

3.3.2 Project Specific Check-sets Structure Definition

The check-sets are a group of different checks that are built to evaluate certain information. The check-sets, therefore, are composed of different individual checks.

Defining a check-set structure that suits the needs of the designer company is important for the following reasons:

- It allows to define a granularity on the checks, which at the same time facilitates the execution of checks to evaluate just certain information without having to execute checks that evaluate the whole BIM models. Therefore, this enhances the principle of modularity for the defined checking activities.
- It increases the control over the information produced.
- It helps the storage of the check-sets in a library, which translates into the possibility of using the same check-sets across projects, or adapted check-sets with limited adaptation efforts.

The **inputs** to perform this activity are:

- Information Requirements suitable to be "rule-implemented" that come as a result of the previous activity of the "Rule Implementation" step of the workflow.

The **controls** for this activity are:

- BIM Authoring Tool (C1): it directly influences the structure of the check-sets, as it would be logical to define a similar hierarchy for the check-sets to the hierarchy used by the Authoring Tool, for example, if the Authoring Tool is Autodesk Revit, it is valid option to structure the check-sets following the same hierarchy (categories, families, types, and instances).

- Project BEP (C2): in particular, it is important to consider the milestones and stages defined for the project and the model breakdown structure.

- Design company Modelling Standards (C3) and Naming Standards (C4).

The **output** of this activity is a "Project Specific Check-Set Structure" that will serve as a control tool for defining the project specific rule-based checks.

To illustrate on how this check-set structure could look like, the next example has been developed:

Table 3: Example of defined Check-Set Structure for the "Rule Interpretation" activity

PROJECT SPECIFIC CHECK-SET STRUCTURE			
DISCIPLINE (Architecture, Structural, Mechanical…)			
	CHECKS BASED ON CATEGORY OF THE OBJECT		
	CATEGORY	CHECKS BASED ON TYPE OR INSTANCE	
		TYPE OR INSTANCE	CONCEPT DESIGN STAGE
			MODELLING CHECKS, CONSISTENCY CHECKS, VERIFICATION CHECKS, VALIDATION CHECKS
			DEVELOPED DESIGN STAGE
			MODELLING CHECKS, CONSISTENCY CHECKS, VERIFICATION CHECKS, VALIDATION CHECKS
			TECHNICAL DESIGN STAGE
			MODELLING CHECKS, CONSISTENCY CHECKS, VERIFICATION CHECKS, VALIDATION CHECKS

3.3.3 Create Project Specific Rule-Based Checks

This activity is the most sensitive of the workflow since it is where the actual creation of rules takes place. The **inputs** for this activity are:

- Suitable Information Requirements that resulted from the Information Requirements Classification activity (A1.1)

- Check-set library (I3): if there are previously defined different checks for other projects and are stored, and those checks have been created following the replicability principle, it is possible that there are available checks in the library that can perform the checking of the desired information with minimum or no adaptation needed. If not, the new checks developed following the replicability, modularity, and storage ability, in this stage should be stored, so they can be used in future projects.

The **controls** for this activity are:

- BIM Authoring Tool (C1): the object-oriented approach of the authoring tool as well as its hierarchy influences the way that the rules are built, as the rules need to identify the elements available in the BIM model, which is usually made by identifying the class of the objects, the type of the objects or the entities.

- Project BEP (C2): as it is defined which and how information in the BIM model is going to be addressed, it directly influences the construction of the project specific checks

- Design Company Modelling Standards (C3) and Naming Standards (C4): these control tools are very important in the creation of the rules. A strong and consistent naming convention is very vital, due to the necessity of identifying the objects and properties available in the model, usually by the object naming or the property naming. In addition, a poor naming convention can hinder the creation of rules, while a good naming convention can reduce exponentially the time needed to replicate the rule. As an example, building a rule with conditions for a rectangular object (such as a beam) that is identified in their name with a code like “Width x Height_Material” (width by height and material) and replicate it for a different element that follows the same naming convention can be a matter of seconds, while if the second rectangular object follows a different naming convention, like “Material_Height”, trying to replicate the same rule can take as much time as writing a new rule from scratch.

The chosen “Rule Translation Tool” (M1) serves as the **mechanism** for carrying out the translation from the requirements to the rule-based checks, in a way that can be read by the “Check Execution Tool” (M3) in the “Rule Execution” step of the workflow. It is common that both the rule translation tool and the check execution tool are the same, allowing to perform both the translation and the execution, but there is also the possibility that the rules can be written in with a different tool (like for example, a text editor that allows to create and edit rules in a language that can be read by the check execution tool, like .xml).

The approach suggested by this workflow is an **object-oriented approach,** as implementing the checks in the BIM Authoring Tool, which commonly follow an object-oriented hierarchy. As opposed of following a software approach like "Solibri Model Checker"[48] could be, which follows a black-box principle in its model checking, it is a matter of importance to use a tool that allows us to follow a **white-box principle**, so the user that creates the rule-based checks can have control on how the information is managed.

To build the rule-based checks, it is proposed that the rule-sets are structured in a way that the following parts can be identified:

- The first part of the rule-set is destined to identify the objects that are intended to be checked (i.e., concrete pile foundations)

- The second part of the rule-set should set the conditions for the alphanumerical information related to the object to satisfy the information requirements (i.e., material of the piles should be concrete)

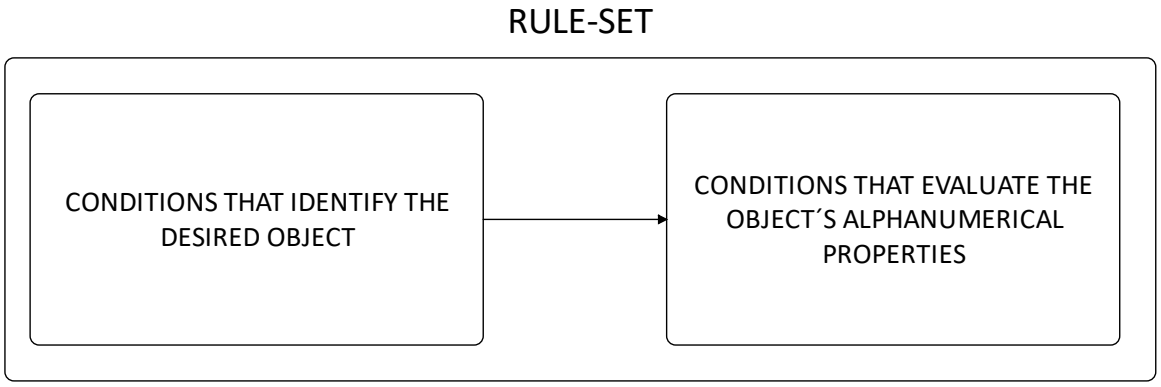

Figure 14: Rule-set Structure

As last step of this activity, it is necessary to decide if after evaluating the rule-set, the check should result in fail or pass, depending on if the rule is built to evaluate the elements that meet the desired conditions or if the rule is built in a way that the elements that are evaluated do not meet the conditions. For example, a check can be built in a way that it identifies the objects that do not have the correct value of a property, and therefore this check returns "Fail" as a result.

As **output** for this activity, the result is the new developed check-sets for the project (different rules conform a rule-set for a check, and checks conform the check-sets) that will evaluate the different information available in the objects of the BIM model.

3.4 Step 2: Building Model Preparation

The activity of "Building Model Preparation" consists of enriching the model with the data referring to the information requirements that it is desired to be checked.

Ideally, this activity should be integrated into the modelling process of the design stages, in order to enhance the efficiency of the project design processes.

For this activity of the process, the **inputs** are:

- The Information Requirements (I1) that served as an input for the "Rule Interpretation" are also input for this step of the process. It is important to say that both Information Requirements that are suitable to be implemented for the automated checking, and the ones that are not suitable to be checked automatically are input for this step, as the model needs to satisfy all the specified requirements.

- Authoring Tool BIM Model (I2) that needs to be enriched with the necessary data to satisfy the information requirements.

The **controls** consist of:

- New developed Check-sets (O1): the developed check-sets that will compare the information in the BIM Model against the rules that are part of the checks during the "Rule Execution" activity.

- Project BEP (C2)

- Design Company Modelling Standards (C3): the model should be modelled and prepared according to the modelling standards of the company for the project. The Modelling Standards are as well a control tool for the previous activity of the "Rule Interpretation".

- Design Company Naming Standards (C4): in the same way that the naming standards are very important in order to develop the rules that are part of the checks, it is obvious that if the model is not prepared using the same naming standards, the developed check-sets will not be able to identify the objects, or the properties embedded in the rules that will perform the checking of the objects in relation to the information requirements.

The BIM Authoring Tool (M2) will be the **mechanism** to perform that Building model preparation, and as a result of this activity, the **output** the output of this activity should be Authoring Tool BIM Model (O2) enriched with data to be checked.

Depending on the maturity level of the design company, this enrichment of the model with data, in particular the alphanumerical information that should be available in the parametric objects, should be carried out by using automated procedures, when possible, in order to ensure that human errors in the process of enriching the model with data are avoided.

To illustrate the above, in the process of modelling during the design stages, the geometrical modelling can be carried out first, and afterwards, the parametric objects that are part of the model can be enriched in an automated manner with the desired properties, and the desired values for these properties can be populated also in an automated way.

Nowadays, there are plug-ins available for the Authoring Tools that allow to create and populate properties for the objects under different standards, like information related to different classification like Uniclass or COBie. One example of this tools is the BIM Interoperability tools for Autodesk Revit [54], that and their Standardized Data Tool or their COBie Extension. There is also a possibility to automate this parametric enrichment through other automation tools like Dynamo.

An example the integration of this activity into a determined design stage modelling process of a project, including automation of the parametric enrichment of the model, can be the following:

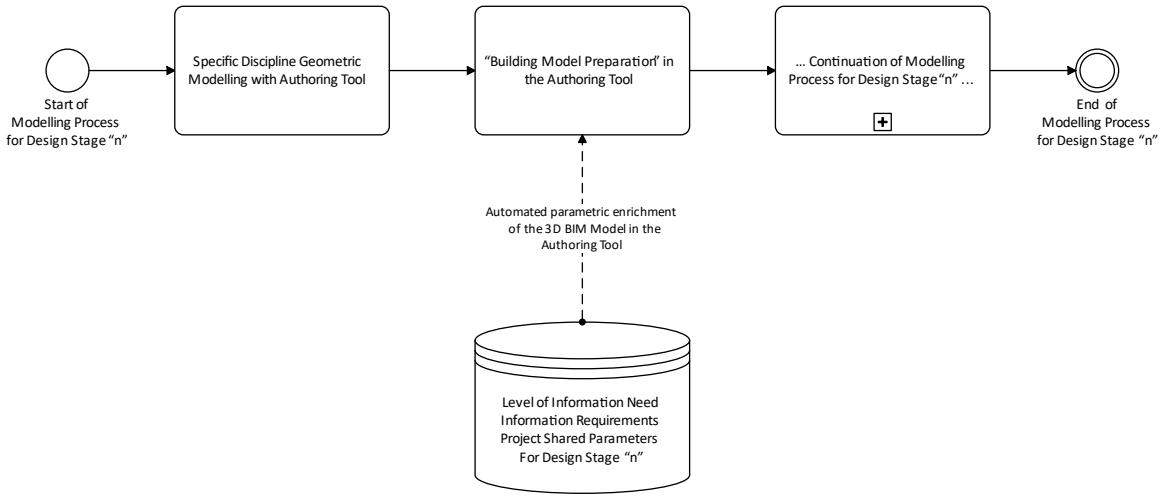

Figure 15: Example of integration of the "Building Model Preparation" activity into a typical modelling process for a determined design stage of a project

3.5 Step 3: Rule Execution

The activity of "Rule Execution" consists in the actual model checking in an automated manner.

The rule execution tool will read the desired data available in the BIM model or the objects that compose it, read the rule-sets that compose the checks, and compare both in order to determine if the information available in the model satisfies the rules implemented in the check at the determined stage of the design process, and therefore the model "passes" the automated check.

Therefore, the **inputs** for this activity are the following:

- Authoring Tool BIM Model (O2): the model that results as an output of the "Building Model Preparation" activity will serve as an input for the "Rule Execution" activity, as the Check-Execution Tool must read the information that is available in the model according to the rules that conform the checks.

- New Developed check-sets (O1): the output of the "Rule Interpretation" activity is also an input for the Check-Execution Tool, as it is needed to indicate which sets of checks it is desired that the tool performs on the information available in the BIM Model.

The **mechanism** for this activity will be the Check Execution Tool (M3) that, as mentioned previously when talking about the "Rule Interpretation" activity, will most likely be the same as the "Rule-Translation" tool, when talking about plug-ins or other software solutions that are designed to perform these automated checks.

As a result of this activity, it will be the results of the checks as an **output**, that will be serving as an input for the "Rule Check Reporting" activity.

The results of the "Rule Execution" will depend on the way that the rule-sets are built. As mentioned before, as part of the "Rule Interpretation" activity, a rule-set can be built in a way that the check should result in "Pass" or "Fail" if the identified conditions for the desired object are matched, which can be useful if it is desired to identify the objects that do not meet the requirements.

In the particular case of this thesis and based on the checking tool used for the case study of the application of this workflow, it has been determined that building the rules in a way that the check fails if the conditions translated into the rule-sets of the checks are met, is of great utility, as identifying the objects that do not comply with the information requirements facilitates the taking of corrective actions on the information associated with the object.

In an illustrative way, this activity takes place in the following way:

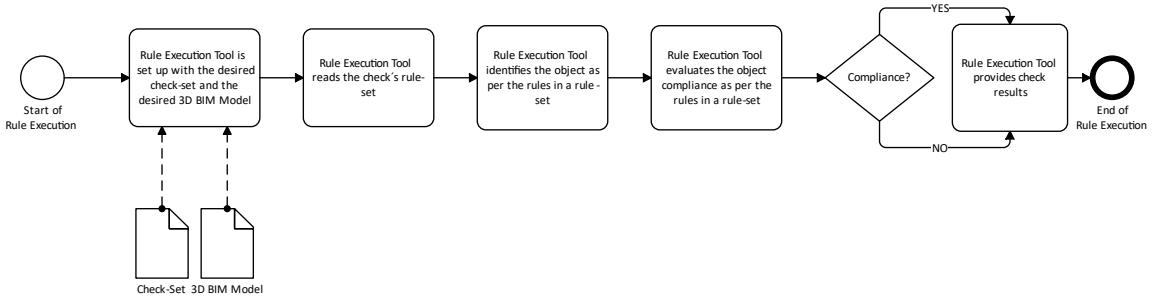

Figure 16: Process of the "Rule Execution" activity

3.6 Step 4: Rule Check Reporting

As the last step of the workflow comes the activity of the Rule Check Reporting, that aims to communicate the results of the performed automated checks to the user that performed such check or other users that may have interest in the information that the results of the checks provide.

As an **input** for the activity, it will have the results of the "Rule Execution" activity, and a **mechanism** to perform the activity will be the Check Results Reporting Tool (M4). The activity will result on the different Check Results Reports generated with the different objectives that the design company estimates as **output.** Generally, when using a software solution such as a plug-in to perform the checks, the tool will have a function to report the results or offer to export the results to a file format, however, if other type of solution is used for the performing of the checks, like a visual programming tool or a scripting tool, it should be functionality should be implemented in a way that the results can be exported in order for them to be visualized or analysed, like for example, to an Excel file.

For the specific case of a design company, once the results of the automatic checks are available, two types of reporting have been identified:

- Information issues reporting: this type of reporting should take place to communicate the results of the checks to the designed project member, so corrective actions can be taken in the BIM model in order to modify the defective information in the model that did not pass the checks. This type of reporting could be carried out using tools that support BCF (BIM Collaboration Format) [55].

- Management reporting: this type of reporting can be prepared by using data visualization tools like Power BI. This type of reporting can serve different objectives, like for example, analyse the trend of the results for a model checked at different delivery milestones, or identifying if

there are some tasks in the modelling process that are not taking place with the expected efficiency.

3.7 Implementation of the workflow in the design process of the project

The defined workflow explains the steps necessary to be taken in order to develop automated rule-based checks for the objective of Model Validation of the BIM model during the different design stages of the project.

However, this workflow should not stand as a parallel workflow to the design process but should be implemented on the project´s design process, with the different steps of the designed workflow happening at different steps of such design process once it is decided to be implemented.

To illustrate this, a generic design process is described in the following diagram, integrating the different steps of the workflow developed in this thesis:

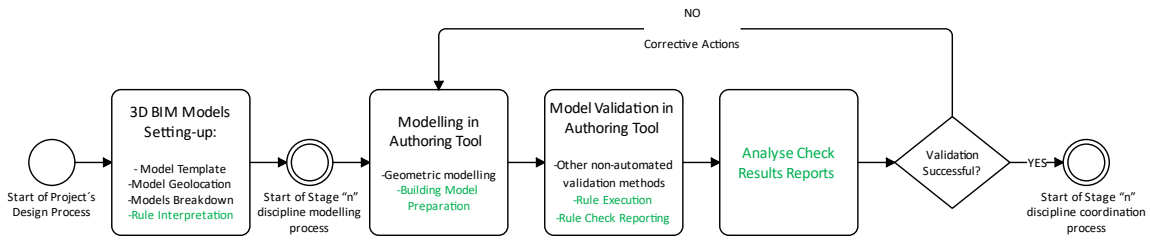

Figure 17: Example of implementation of the workflow steps into a project generic design stage

4 CASE STUDY

4.1 Partner Institution and case study material

The material used for this case study has been facilitated by IBE d.d.. They are the leading engineering and consulting company in Slovenia, with more than 70 years of experience, and they develop projects in different sectors such as energy, industry, infrastructure, public service buildings or environmental protection, among others.

With more than 160 employees located in different offices across the country, the company has taken a step forward in the adoption of BIM Methodology on their daily basis. Using different authoring tools for their projects, one of the most used for this is Autodesk Revit for the development of BIM models for disciplines such as architecture, structures, and MEP.

For the development of this thesis, IBE d.d. has provided the author with a BIM model developed with Autodesk Revit. The model consists in the Structural Model developed at the Technical Design stage for the project of a logistics warehouse for one of their clients. It is worth mentioning that the model is a "snapshot" (which means the model is frozen in time) of the BIM model, ready to be used independently and containing three-dimensional information and alphanumerical information for the objects that comprise the BIM model. An IFC and NWD files with the federated model of the project has also been facilitated to give context to the information.

Figure 18: Decomposed view of the building model for the logistics warehouse project.

Figure 19: Building model for the logistics warehouse project.

In addition to the BIM model, IBE d.d. has also provided the list of parameters and values for those, required for the **structural elements of the foundations of the building (which will be the scope of the case study)**. The list of parameters and values can be identified as the "Alphanumerical Information" that is part of the Level of Information Need required and are identified for the different design stages of the project (Faze 1 corresponds to "Concept Design", Faze 2 to "Developed Design", and Faze 3 to "Technical Design"). The list has been adapted to the facilitated model from a most recent project currently under development by the company in an earlier stage of the design phase of the life cycle of the project.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Id	Tip Elementa	Skupina atributov	Atribut - IFC	Vrsta vnosa	Atribut - izvorni	Enota	Primer vnosa	Komentar	PZI faze		
2										Faza 1	Faza 2	Faza 3
3	G-01-001	Pilot	IBE_Klasifikacija	ElementTip	Text	IBE_ElementType	/	Pilot - uvrtan	/		X	X
4	G-01-002	Pilot	IBE_Identifikacija	Oznaka	Text	Mark	/	PIL-K1-001	/		X	X
5	G-01-003	Pilot	IBE_Identifikacija	Opis	Text	Description	/	Uvrtan pilot D=100 cm, L=25 m	Opisno		X	X
6	G-01-004	Pilot	IBE_Identifikacija	Komentar	Text	Comments	/	<komentar>	Opcijsko			
7	G-01-005	Pilot	IBE_Identifikacija	OznakaTip	Text	Type Comments	/	PIL-D100-A	/		X	X
8	G-01-006	Pilot	IBE_Identifikacija	KomentarTip	Text	Type Mark	/	D100	/		X	X
9	G-01-007	Pilot	IBE_IfcParametri	IfcExportAs	Text	IfcExportAs	/	IfcPile	Določeno	X	X	X
10	G-01-008	Pilot	IBE_IfcParametri	IfcExportType	Text	IfcExportType	/	BORED	Določeno	X	X	X
11	G-01-009	Pilot	IBE_Dimenzije	Premer	Length	Diameter	[m]	1 m	/	X	X	X
12	G-01-010	Pilot	IBE_Dimenzije	Dolzina	Length	Length	[m]	25 m	/	X	X	X
13	G-01-011	Pilot	IBE_Dimenzije	KotaVrha	Length	Elevation at Top	[m]	-8 m	/	X	X	X
14	G-01-012	Pilot	IBE_Dimenzije	KotaDna	Length	Elevation at Bottom	[m]	-30 m	/	X	X	X
15	G-01-013	Pilot	IBE_Material	Material	Text	Structural Material	/	Beton-C25/30	/		X	X
16	G-01-014	Pilot	IBE_Material	BetonRazred	Text	IBE_ConcreteGrade	/	C25/30	/		X	X
17	G-01-015	Pilot	IBE_Material	Razredizpostavljenosti	Text	IBE_ExposureClass	/	XC0	/			X
18	G-01-016	Pilot	IBE_Material	RazredKloridov	Text	IBE_ChlorideClass	/	Cl 0.2	/			X
19	G-01-017	Pilot	IBE_Material	MaxPremerZrna	Text	IBE_ConcreteDmax	/	D16	/			X
20	G-01-018	Pilot	IBE_Material	RazredKonsistence	Text	IBE_ConcreteSlumpClass	/	S3	/			X
21	G-01-019	Pilot	IBE_Konstrukcijski	Structural	Boolean	Structural	/	Yes	/			X
22	G-01-020	Pilot	IBE_Konstrukcijski	ArmaturaRazred	Text	IBE_ReinforcementGrade	/	B500B	/			X
23	G-01-021	Pilot	IBE_Konstrukcijski	ArmaturaDelez	Integer	IBE_ReinforcementRatio	[kg/m3]	80	/			X
24	G-01-022	Pilot	IBE_Konstrukcijski	ZascitniSloj	Length	IBE_RebarCover	[m]	0.04	/			X
25	G-01-023	Pilot	IBE_Konstrukcijski	Nosilnost	Force	IBE_PileBearingCapacity	[kN]	7.0 kN	/			X
26	G-01-024	Pilot	IBE_Izvedba	Izvedba	Text	IBE_ConstructionProcess	/	InSitu	/		X	X
27	G-01-025	Pilot	IBE_Izvedba	Razredizvedbe	Text	IBE_ExecutionClass	/	EXC2	/			X
28	G-01-026	Pilot	IBE_Izvedba	RazredPovrsinskeObdelave	Text	IBE_SurfaceTreatment	/	VB0	/			X
29	G-01-027	Pilot	IBE_Izvedba	RazredNege	Text	IBE_CuringClass	/	1	/			X
30	G-01-028	Pilot	IBE_Izvedba	RazredTolerance	Text	IBE_ToleranceClass	/	1	/			X
31	G-01-029	Pilot	IBE_Izvedba	OpazTip	Text	IBE_SideFormworkType	/	JeklenaCev	/			X
32	G-01-030	Pilot	IBE_Lokacija	Nivo	Text	Level	/	K1	/	X	X	X
33	G-01-031	Pilot	IBE_Lokacija	LokacijaOsi	Text	IBE_LocationMark	/	A(+2,6)1	+ pomeni gor/desno	X	X	X
34	G-01-032	Pilot	IBE_Lokacija	KoordinataX	Length	IBE_CoordinateX	[m]	15 m	/		X	X
35	G-01-033	Pilot	IBE_Lokacija	KoordinataY	Length	IBE_CoordinateY	[m]	15 m	/		X	X
36	G-01-034	Pilot	IBE_Popis	ReferencnaStevilkaIZS	Text	Keynote	/	01.07.01.XXX	Glej sistemizacijo IZS			X
37	G-01-035	Pilot	IBE_Popis	OpisPostavkeIZS	Text	IBE_OpisPostavkeIZS	/	Uvrtan pilot D=100 cm, C25/30	Glej sistemizacijo IZS			X
38	G-01-036	Pilot	IBE_Popis	Volumen	Volume	IBE_Volume	[m3]	19,6 m3	/			X
39	G-01-037	Pilot	IBE_Popis	PovrsinaPlasc	Area	IBE_PileShaftArea	[m2]	78,5 m2	/			X

Pilot | PilotnaGreda | PodlozniBeton | TemeljnaPlosca | TockovniTemelj | PasovniTemelj | TemeljnaCasa | Skupine

Figure 20: Example of alphanumerical information required for the pile foundation elements of the model depending on the design stage of the project

4.2 Software selection

For the practical application of the methodology described in the previous chapter, **Autodesk Revit 2022** has been chosen as the Authoring Tool, for the obvious reasons that the partner institution IBE d.d. develops their projects with this tool.

As for the tools to carry out the activities for automated rule-based checks, the tools that have been chosen are the following:

- Autodesk Model Checker Configurator [56]: this tool will be used as the "Rule Translation Tool" specified in the methodology, and allows to create custom rule-based checks that can be later executed in an automated way.

- Autodesk Model Checker for Revit [57]: this will be used as the "Check Execution Tool" specified in the methodology, and allows to execute the rule-based checks created with the configurator on the authoring tool BIM model.

- Dynamo [58]: this visual programming tool for Autodesk Revit can be used for many applications. In the case of the particular application of the defined methodology, this tool has been used to populate parameters with the required values defined in the alphanumeric information requirements from the Level of Information Need. It will be used to populate the values for the parameters that are missing in the model during the activity of Building Model Preparation, in an automated way.

- Power BI [59]: this tool has been used to create management reports to help understand the checked information and the outcome of the checks. It will be used as the "Check Results Reporting Tool" mentioned in the methodology, to illustrate how the results of the checks can be reported.

4.3 Methodology application: Step 1 – Rule interpretation

4.3.1 Information Requirements Classification

As described in the methodology chapter, this is the first activity for the "Rule Interpretation" step, where the requirements are classified into the ones that that can be translated into machine-readable rules, and those that cannot.

Due to both confidentiality and language barriers, it has not been possible to have access to all the requirements for the project (OIR, PIR, EIR, etc…). In any case, **the list of alphanumerical information requirements has been provided,** which are all full machine-readable and are suitable to

be implemented into the BIM model. This list of alphanumerical information requirements is part of the Level of Information Need required and will be used as the input for the next activities of the first step of the workflow. The list of alphanumerical information required can be consulted in the "Annex 1: Alphanumerical Information Requirements".

4.3.2 Project Specific Check-sets Structure Definition

The aim of this activity is to define a correct structuring of the check-sets that will be developed, in a way that the adjust to the necessities of the model checking activity.

As the scope of the case study focuses on the "Structural Foundations" elements that are part of the structure of the building. The elements of this category which alphanumerical information required has been provided and that will be checked are:

- Piles
- Pile Caps
- Ground Beams
- Pocket Foundations
- Foundation Slabs

In addition to the identification of the elements that will be checked, a classification on the types of checks to be performed can also be made. For this case study, the checks have been classified into 4 different types of checks:

- Modelling checks (MOD): a small series of basic modelling checks have been developed based on correct modelling practices. For example, the elements are associated to a level. These checks are also related to **active properties** of the objects.

- Consistency Checks (CON): the aim of this checks is to ensure that the information provided by the naming of the elements is consistent with the geometrical information or alphanumerical information. For example, the information for the width on the "type name" of an object is consistent with the actual width dimension of the object.

- Verification Checks (VE): These checks aim to verify the existence of the information parameters required by the Level of Information Need in the objects, provided by IBE. They do not check if the value of the parameter is correct or not. For example, the existence of the parameter IBE_ElementType.

- Validation Checks (VA): the objective of this set of checks is to evaluate the value of the parameters against the required values and, therefore, validate the information content required by the "Level of Information Need" provided by IBE. For example, if the value of the parameter IBE_ElementType is correct for the piles. These checks are performed over **passive properties** of the objects.

The grouping of the alphanumerical information provided by the partner institution has also been taken into account for the definition of the structure. The provided parameters are classified in the following groups:

- Classification / IBE_Klasifikacija
- Identification / IBE_Identifikacija
- IFC Parameters / IBE_IfcParametri
- Dimension / IBE_Dimenzije
- Material / IBE_Material
- Structural / IBE_Konstrukcijski
- Construction / IBE_Izvedba
- Location / IBE_Lokacija
- Description / IBE_Popis

As a result, the check-sets developed have been structured in a way that allows modularity of checking, considering the type of checks developed. This structure will allow to select which checks are wanted to be executed, without having to execute all of them at the same time, if this was not necessary. For example, it can be chosen to execute just checks related to the ground beams (modelling, consistency, verification, and validation) or execute just a type of check for all the structural foundation elements. The project specific check-set structure is illustrated in the following table:

Table 4: Defined project specific check-set structure

STRUCTURAL INFORMATION CHECKS			
STRUCTURAL FOUNDATIONS: GENERAL			
	SFGeneral	Verification of Parameters (VE)	
		SFGeneral (VE)	Classification / IBE_Klasifikacija IFC Parameters / IBE_IfcParametri Material / IBE_Material Structural / IBE_Konstrukcijski Construction / IBE_Izvedba Description / IBE_Popis
STRUCTURAL FOUNDATIONS: PILES			
	Piles	Modelling Procedures (MOD)	
		Consistency (CON)	
		Verification of Parameters (VE)	
		Piles (VE)	Structural / IBE_Konstrukcijski Location / IBE_Lokacija
		Validation of Parameters (VA)	
		Piles (VA)	Classification / IBE_Klasifikacija Identification / IBE_Identifikacija IFC Parameters / IBE_IfcParametri Material / IBE_Material Structural / IBE_Konstrukcijski Location / IBE_Lokacija Construction / IBE_Izvedba Description / IBE_Popis
STRUCTURAL FOUNDATIONS: PILECAPS			
	Pile Caps	Modelling Procedures (MOD)	
		Consistency (CON)	
		Validation of Parameters (VA)	
		Pile Caps (VA)	Classification / IBE_Klasifikacija Identification / IBE_Identifikacija IFC Parameters / IBE_IfcParametri Material / IBE_Material Structural / IBE_Konstrukcijski Construction / IBE_Izvedba Description / IBE_Popis
STRUCTURAL FOUNDATIONS: GROUND BEAMS			
	Ground Beams	Modelling Procedures (MOD)	
		Consistency (CON)	
		Validation of Parameters (VA)	
		Ground Beams (VA)	Classification / IBE_Klasifikacija Identification / IBE_Identifikacija IFC Parameters / IBE_IfcParametri Material / IBE_Material Structural / IBE_Konstrukcijski Construction / IBE_Izvedba Description / IBE_Popis

STRUCTURAL FOUNDATIONS: POCKET FOUNDATIONS			
	Pocket Foundations	Modelling Procedures (MOD)	
		Consistency (CON)	
		Validation of Parameters (VA)	
		Pocket Foundations (VA)	Classification / IBE_Klasifikacija
			Identification / IBE_Identifikacija
			IFC Parameters / IBE_IfcParametri
			Material / IBE_Material
			Structural / IBE_Konstrukcijski
			Construction / IBE_Izvedba
			Description / IBE_Popis
STRUCTURAL FOUNDATIONS: FOUNDATION SLABS			
	Foundation Slabs	Modelling Procedures (MOD)	
		Consistency (CON)	
		Validation of Parameters (VA)	
		Foundation Slabs (VA)	Classification / IBE_Klasifikacija
			Identification / IBE_Identifikacija
			IFC Parameters / IBE_IfcParametri
			Material / IBE_Material
			Structural / IBE_Konstrukcijski
			Construction / IBE_Izvedba
			Description / IBE_Popis

Once the structure is defined, the next step is to build the check-set structure on the tool that will be used to build the rule-based check-sets. As mentioned before, for the development of this thesis, the tool selected for this activity is Autodesk Model Checker Configurator from the Autodesk BIM Interoperability Tools.

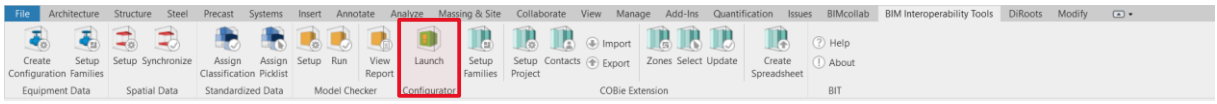

Figure 21: Autodesk Model Checker Configurator launcher in Revit 2022.

Once the tool is launched, it is possible to create a new check-set file from scratch or create a new one by editing an existing one that can be in the company´s library, which could be used as a template to create different project specific check-sets. The idea of a library is one of the cases where the modularity principle becomes essential. Because of the way the Model Check configurator works, importing individual checks is not possible. However, it is possible to have individual checks under the category of "Unused Checks", which can act as a library.

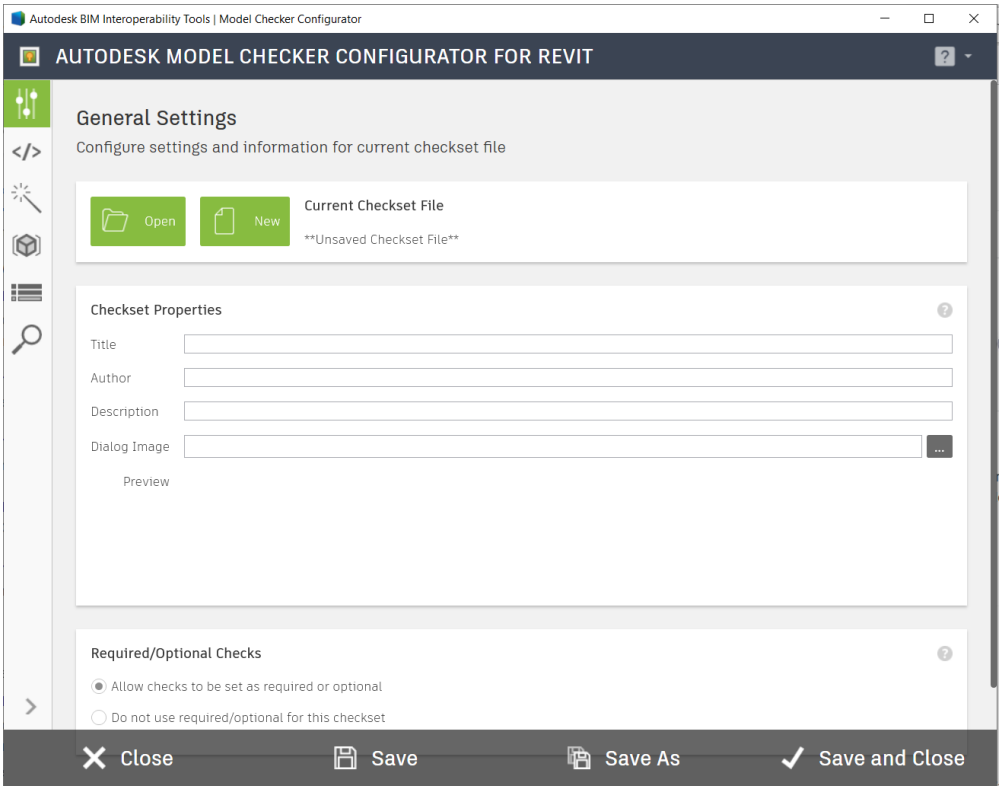

Figure 22: Autodesk Model Checker Configurator. General Settings interface

In the case of this thesis, a new check-set has been created from scratch for the case study. The structure of the check-set can be defined under the option of "Checkset Structure and Organization"

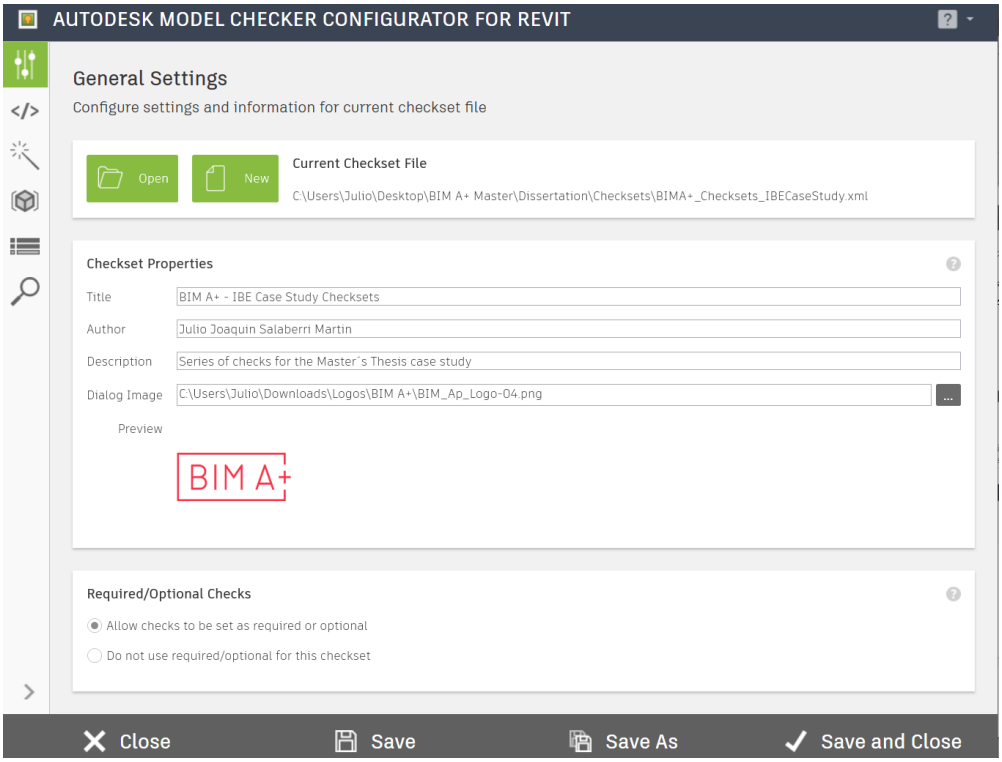

Figure 23: Setting up of the case study check-set information

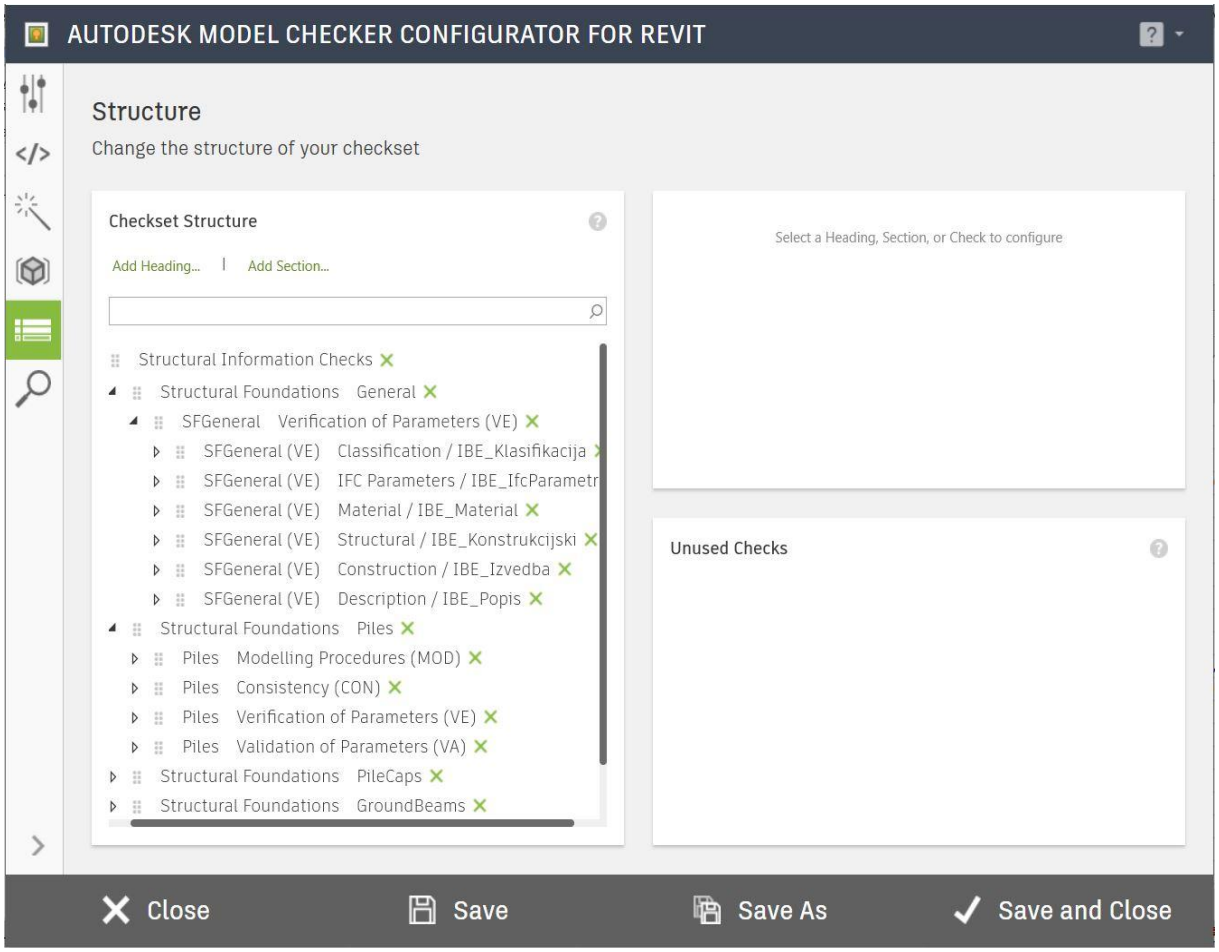

Figure 24: Setting up of the project specific check-set structure with the Configurator

4.3.3 Create Project Specific Rule-Based Checks

Once the check-set structure is defined, and the requirements against which it is desired to check the BIM model´s objects are identified, the next step is to build the different checks that will validate the model´s alphanumerical information content.

To build these checks, the same tool, Autodesk Model Checker Configurator, will be used. This tool allows to create rule-based checks in three different ways:

1. The “Wizard Check Builder” allows to create checks in an interactive way, choosing between the different options that are prompted in the screen, from the result that is desired for the check, through the elements to be checked, to the properties that should be checked.

2. The “Pre-Built Checks” option allows to select from a set of standard checks that come already built by the tool and that can be used to check general aspects of the model.

3. The “Advanced Check Builder” allows to create new checks from scratch or edit existing ones by editing the rules (filters) that form the check. These checks can be later added to the corresponding place under the check-set structure. This is the option that has been chosen to start creating the rule-based checks.

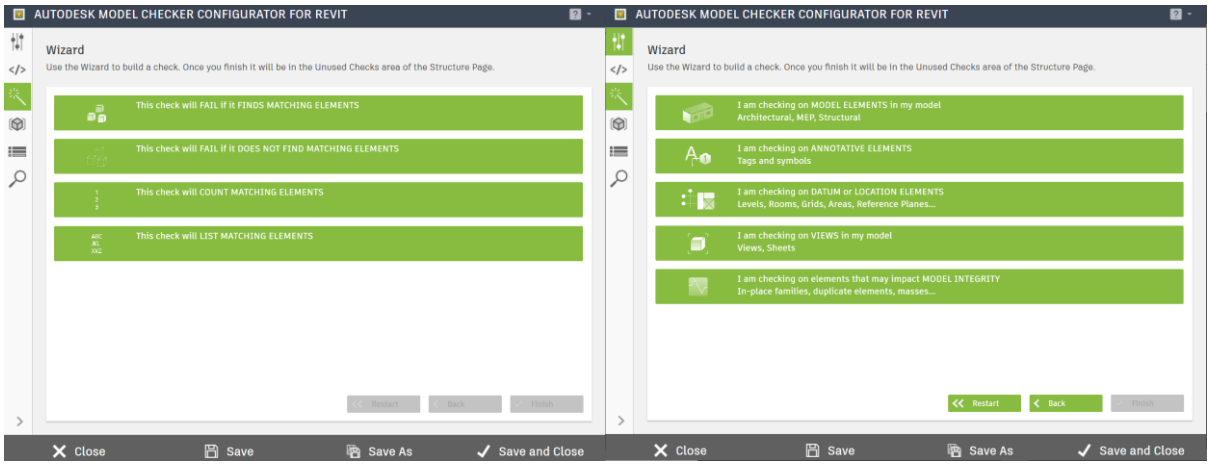

Figure 25: “Wizard Check Builder” interface of the Model Checker Configurator

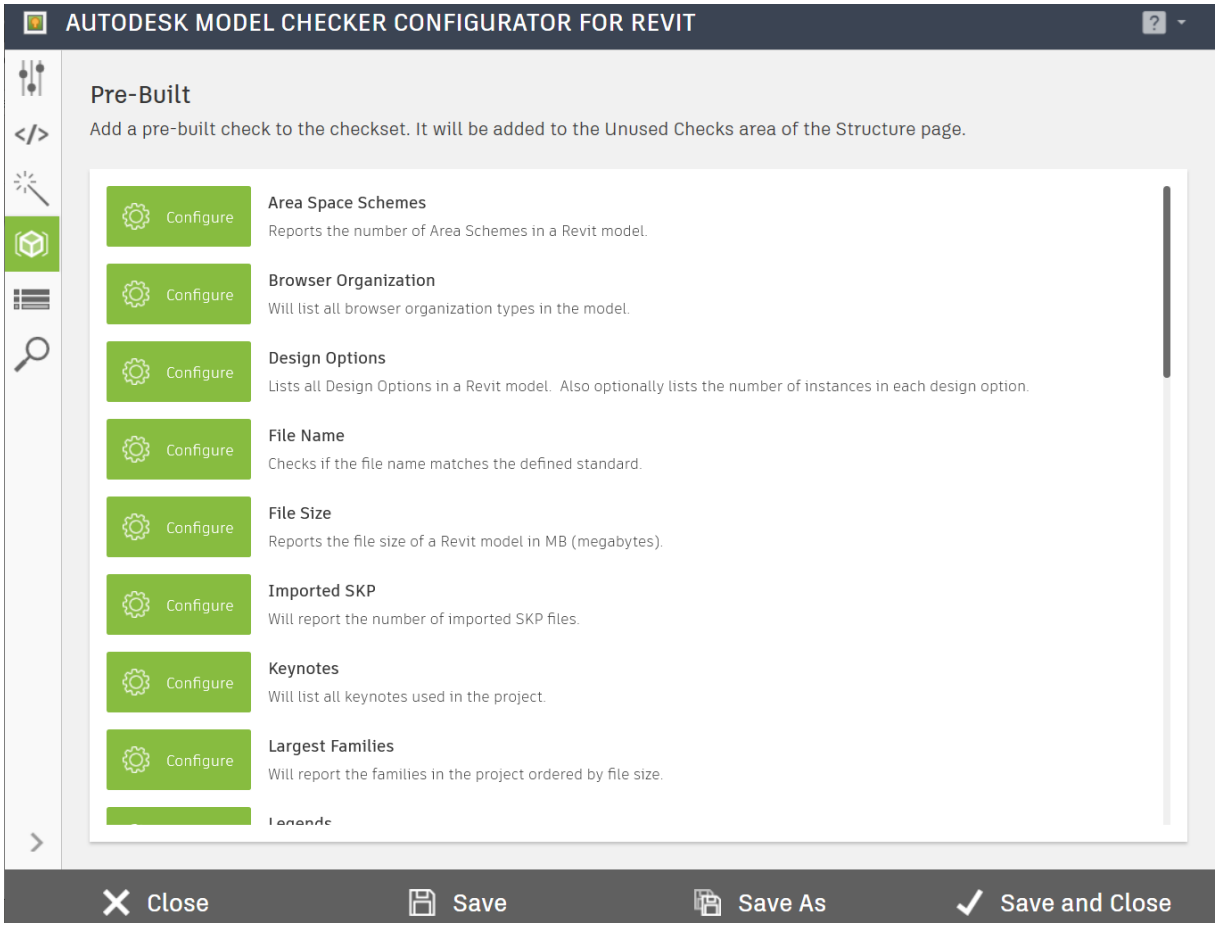

Figure 26: “Pre-Built Checks” interface of the Model Checker Configurator

When creating checks from scratch with the "Advanced Check Builder", the rules that will be part of the check can be built with what the tool calls as "filters". Each filter is composed of:

- Operator
- Criteria
- Property
- Condition
- Value

To follow the rule-set structure defined in the methodology (refer to Figure 14) the combination of different filters needs to be used, some of them to set the conditions that identify the desired objects, and some of them will be used to evaluate the object´s properties against the requirements.

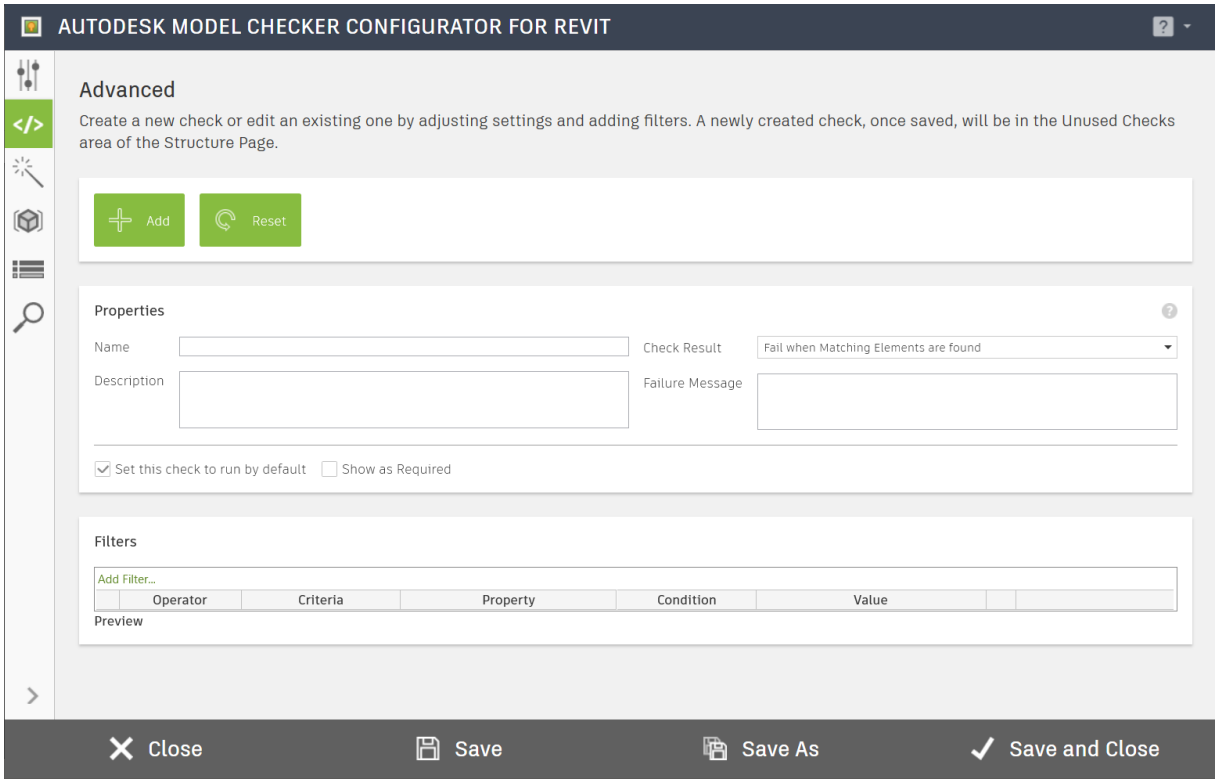

Figure 27: "Advanced Check Builder" interface of the Model Checker Configurator

When building the checks with the "Advanced Check Builder", four different options for the result of the check can be chosen:

- Fail when No Matching Elements are found
- Fail when Matching Elements are found
- Count of Matching Elements only
- Count and list of Matching Elements

The choice of the "Check Result" for the check is important because it influences the logic of the filters (rules) that will be used to compose the check.

The different checks for the present case study developed with the Configurator tool have been built in a way that the checks will fail when matching elements are found. This has been decided because, due to the characteristics of the tool, when the check fails due to matching elements being found, the tool will report a list of the matching elements, this means, the elements that are failing the check. This is useful to identify the objects that are not being compliant with the requirements.

Considering the characteristics of the Configurator tool, the individual checks have been configured following the next structure:

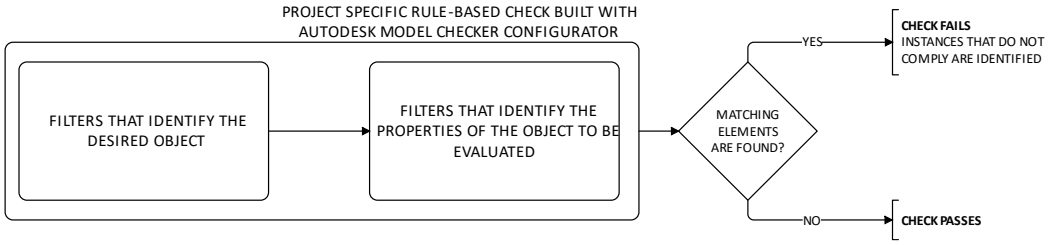

Figure 28: Project specific rule-based structure for the case study

To illustrate, in the next example it can identified how a project specific rule-based check is built for this case study:

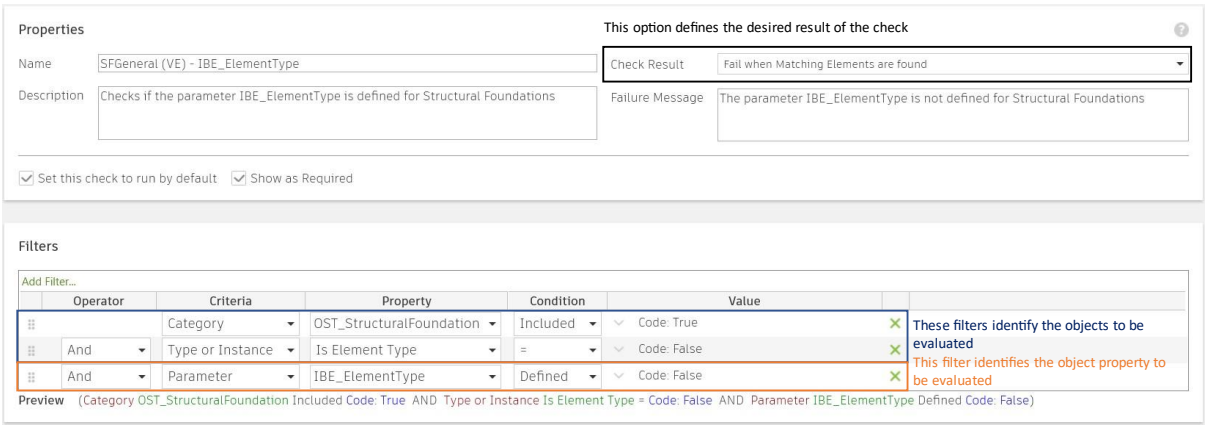

Figure 29: Example of project-specific check developed with the Model Checker Configurator for the verification of the parameter IBE_ElementType of the "Structural Foundation" elements.

- The desired result is the check to "Fail when Matching Elements are found".
- The combination of the first two filters, identify the objects that are desired to be evaluated. In this case, all the instances withing the category "Structural Foundations".

- The third filter identifies the property of the objects to be evaluated objects against the requirements. For this case, the requirement is the existence of the parameter "IBE_ElementType".

As for the **granularity of the checks** built for this case study, the checks can be built in a way where they can check a single requirement at a time or different requirements at the same time. Having more a higher granularity translates into the creation of a bigger number of checks but, given the internal structure of the tool and how it organizes the filters to build the checks, having a lower granularity does not mean that the checks are faster to be created.

For this case study, a higher granularity on building the checks has been applied, in a way that each check checks just a single requirement, which helps to identify in a better way which objects are not compliant.

It is mentioning that the check-set created will be an .xml file. This means, that the file cannot be only opened or edited with the Model Checker Configurator, but it can also be opened and edited with different tools like text editors or source-code editors.

The different checks created for the case study have been created to ensure that the replicability of the checks is possible. In order to corroborate this, the first checks, corresponding to the pile instances were created with the Autodesk Model Checker Configurator, which took a certain amount of time since all of them had to be created from scratch. Once this check-set for the pile foundation instances was created, it has been replicated for the rest of the elements (pile caps, ground beams, pocket foundations and foundation slabs). For this, the .xml format has resulted in a tangible amount of time saved due to the possibility of opening it with a source-code editor and copy/paste the checks that were already created and edit the new ones with tools like find/replace so they will check the new desired elements.

After editing the file with the source-code editor, the file can be opened again with the Model Check Configurator, which will recognize all the new checks present in the file.

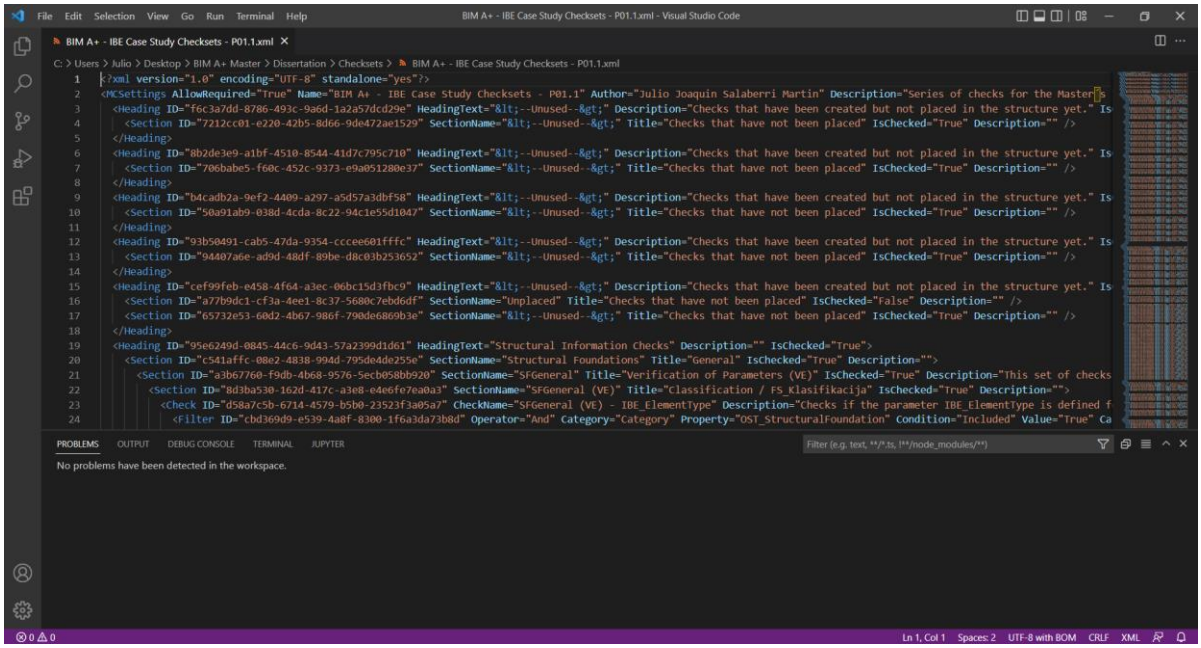

Figure 30: Check-set file opened in a source code editor

Once the check-set has been defined and structured in the desired way and all the desired checks have been created, it is possible to proceed to the next step of the workflow. The full list of project specific checks inside the check-sets can be found in the “Annex 2: Project Specific Check-set”.

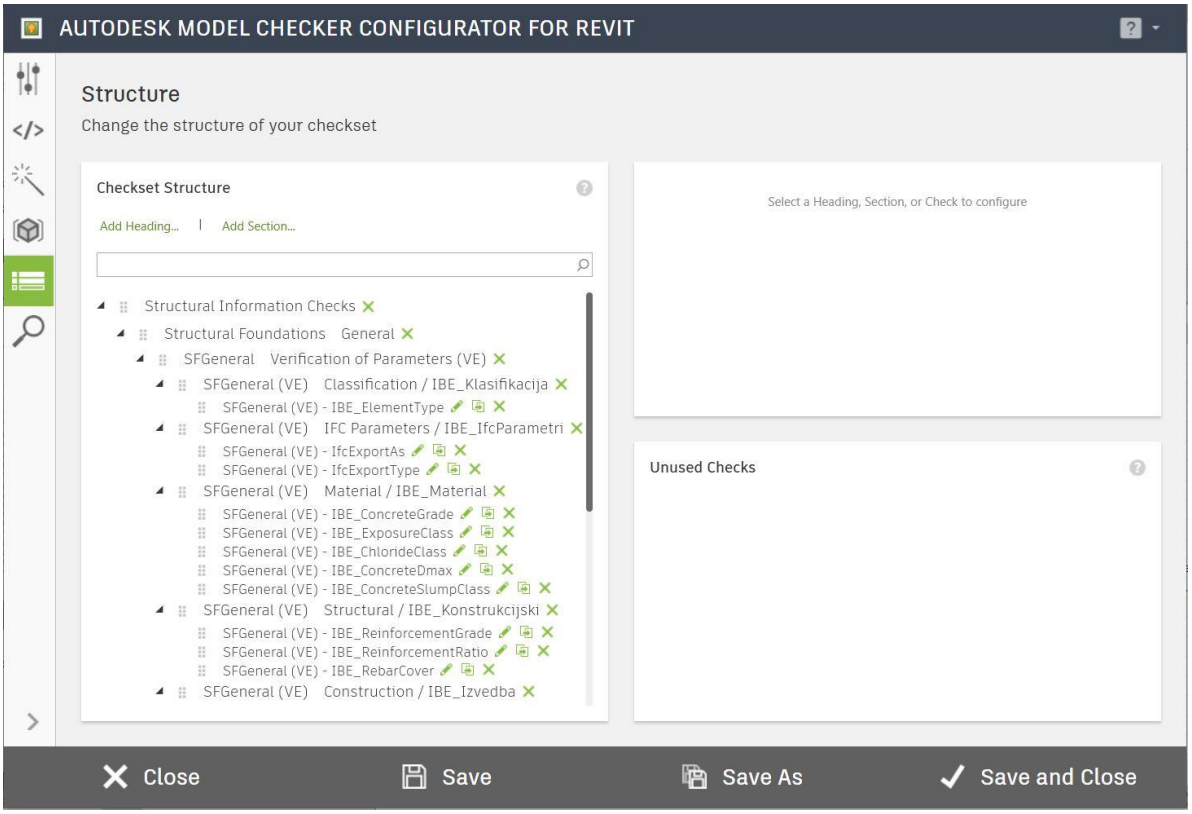

Figure 31: Check-set with the project specific rule-based checks created for the case study in the shown in the Configurator interface

4.4 Methodology application: Step 2 – Building Model Preparation

As mentioned during the methodology chapter of this thesis, this activity should take place integrated during the modelling process of the design stages of the project, and not as an independent activity to take place after such process.

In addition, it has also been mentioned the potential of preparing on enriching the model with data using automated procedures in order to eliminate or reduce human errors when preparing the model.

For the development of this case study, it has been assumed that the model provided has been already prepared before being provided, although, it should be mentioned that the requirements provided were not originally required for the provided project, and therefore some data might be missing (this will, after, portrait a more realistic view of the check execution and the results of the checking).

Some of that missing data mentioned has been identified during the application process of the methodology to the case study. In particular, the absence of the following parameters in the structural foundation elements:

- IBE_ConcreteGrade
- IBE_ExposureClass
- IBE_ChlorideClass
- IBE_ConcreteDmax
- IBE_ConcreteSlumpClass

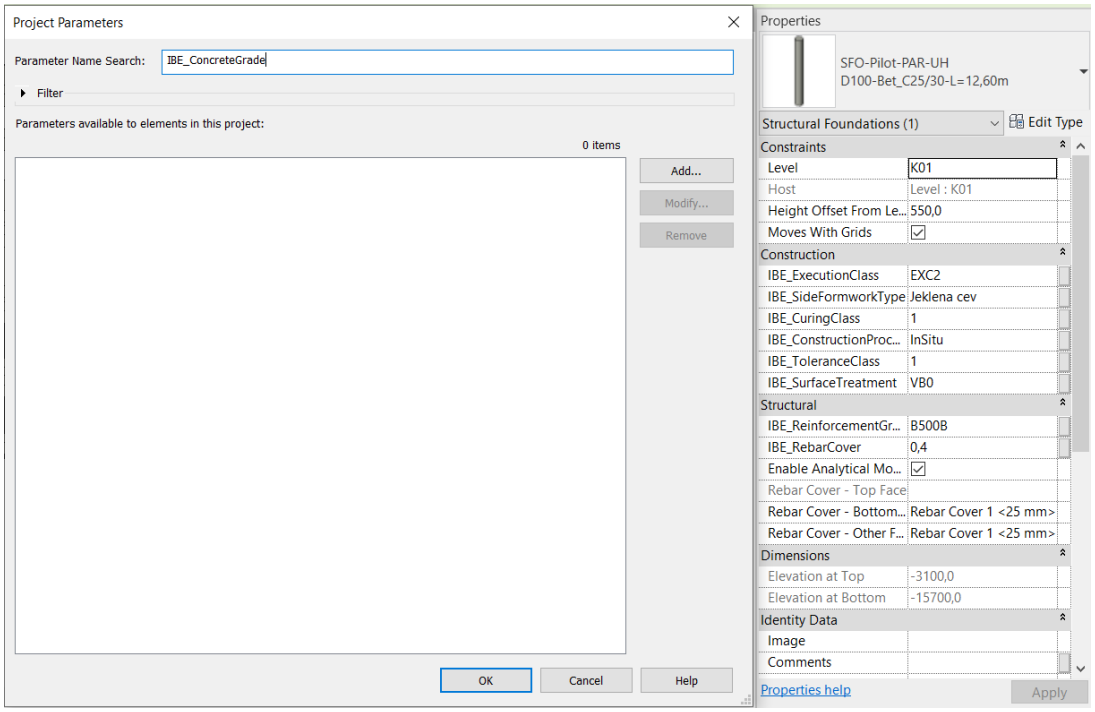

Figure 32: Parameter “IBE_ConcreteGrade” missing in the structural foundation elements

The fact that this data is missing in the model has been perfect to allow to illustrate how the enrichment of the BIM model can be done in an automated way. For this, a Dynamo script has been built, allowing to create the missing parameters and populate the parameters with the value required in the provided list of alphanumerical information required. It is needed to mention that the following script creates new parameters instead of adding parameters from the design company shared parameters file (as it was not possible to access to it), but a script that adds existing parameters from a shared parameters file could also be built, so the logic behind the automated enrichment of the alphanumerical information of the model is still the same, and therefore, valid.

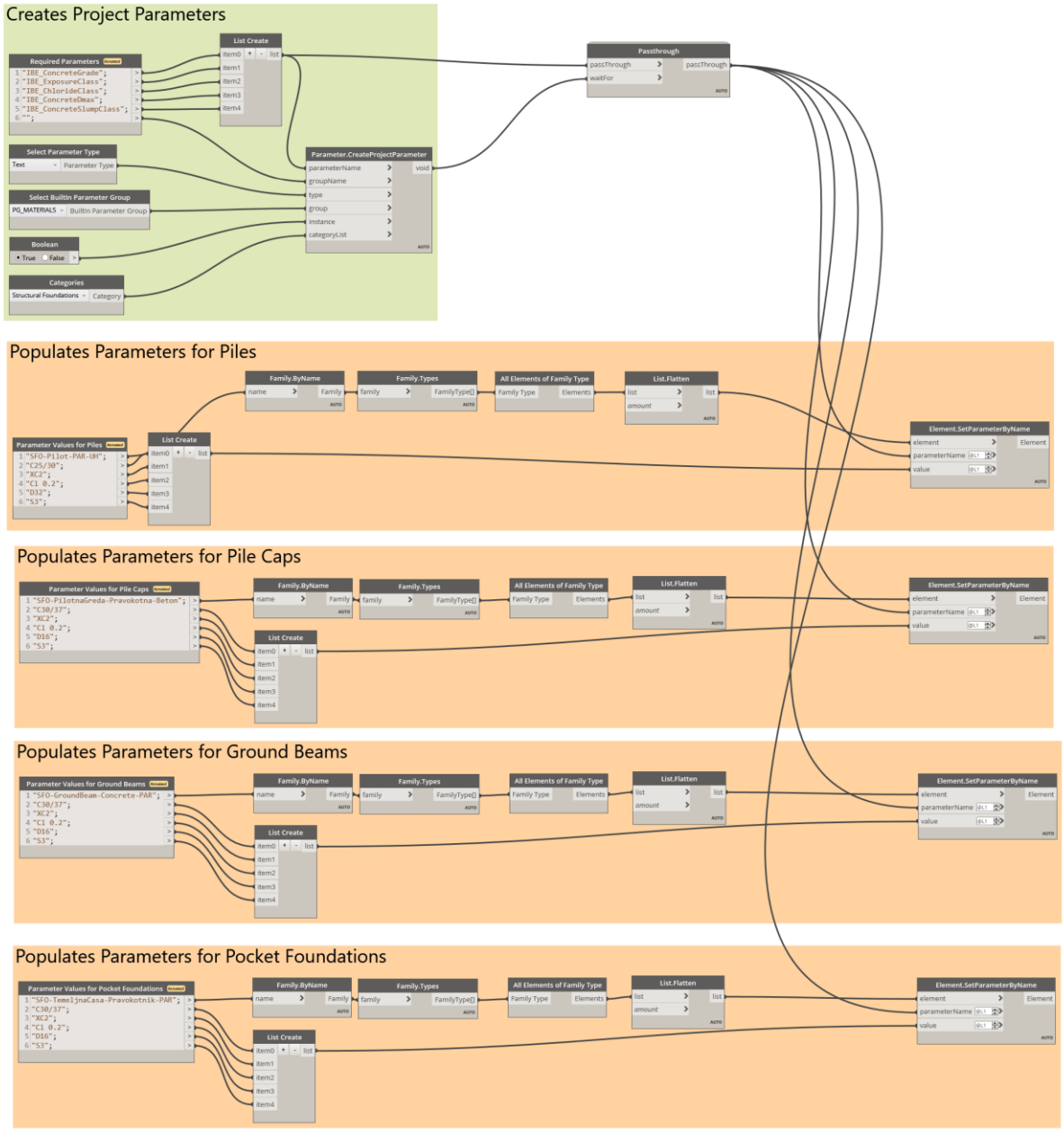

Figure 33: Dynamo script to for model data enrichment in an automated manner

The script has been built in a way that both the creation of the parameters and the population of the values happen sequentially by introducing a pass-through node. This allows to have just one script instead of two different scripts, one for creating the parameters and other to populate the value of the parameters. Once the script is in the library of the company, it can be executed with the Dynamo Player tool, without the necessity of opening the script editor. After the script is executed, the parameters will be available in the objects and populated with the desired value, as can be seen in the following image:

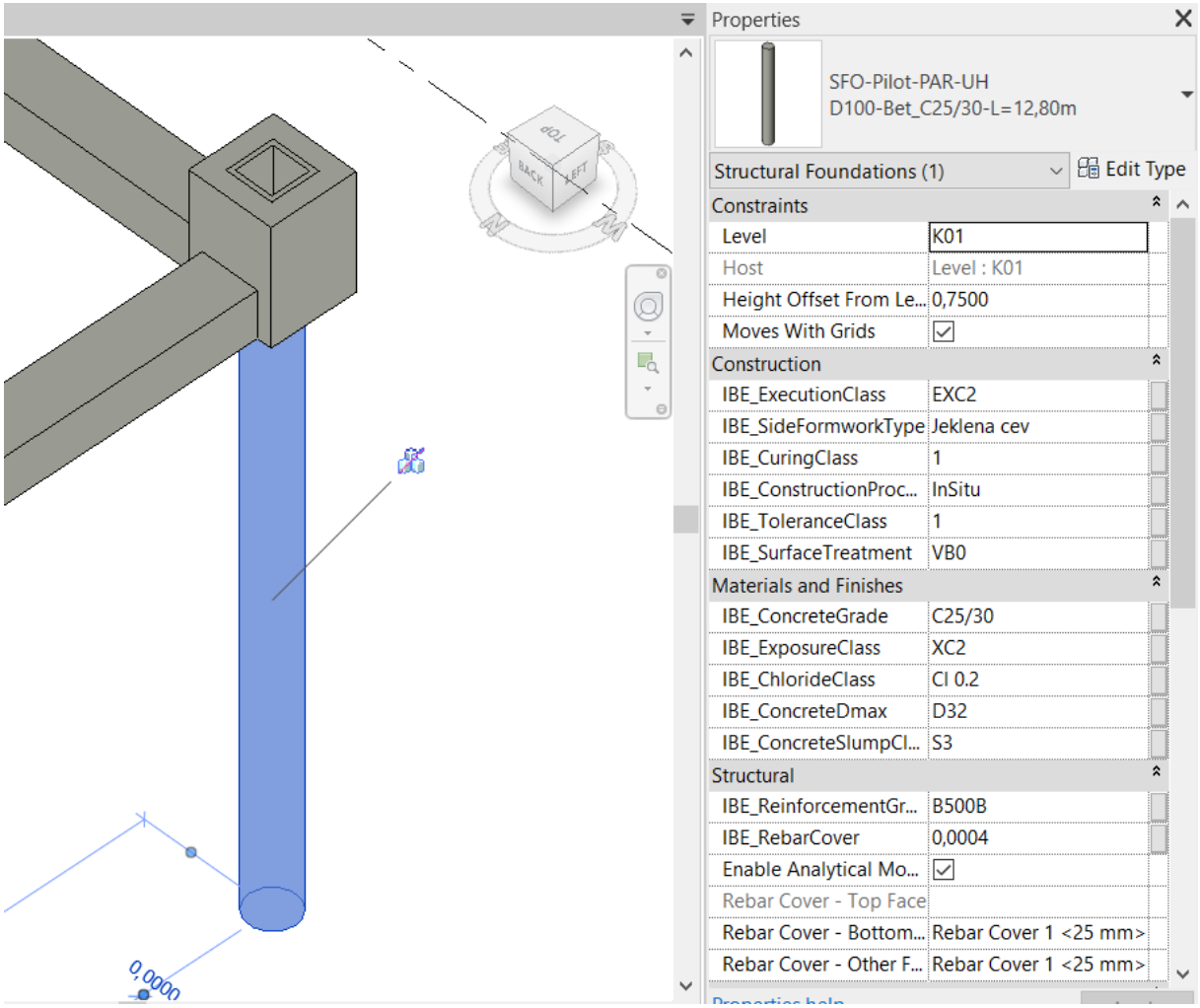

Figure 34: Missing parameters available and populated for the elements of the model

Ideally, the enrichment of the alphanumerical information of the objects in the BIM Model should be done at every design stage by the design team, just with the minimum information needed to satisfy the alphanumerical information requirements, and therefore, avoiding waste.

4.5 Methodology application: Step 3 – Rule Execution

This step of the workflow is a sensitive one, as generally the execution of the checks needs to be triggered by a human. However, ideally, this step could happen in an automated way, automated meaning that it does not need human activity.

For the case study, the "Check Execution Tool" has been the Autodesk Model Checker for Revit. To execute checks with this tool, it is first necessary to setup the check-set file, this means, associate the check-set file to the BIM model that is wished to be checked:

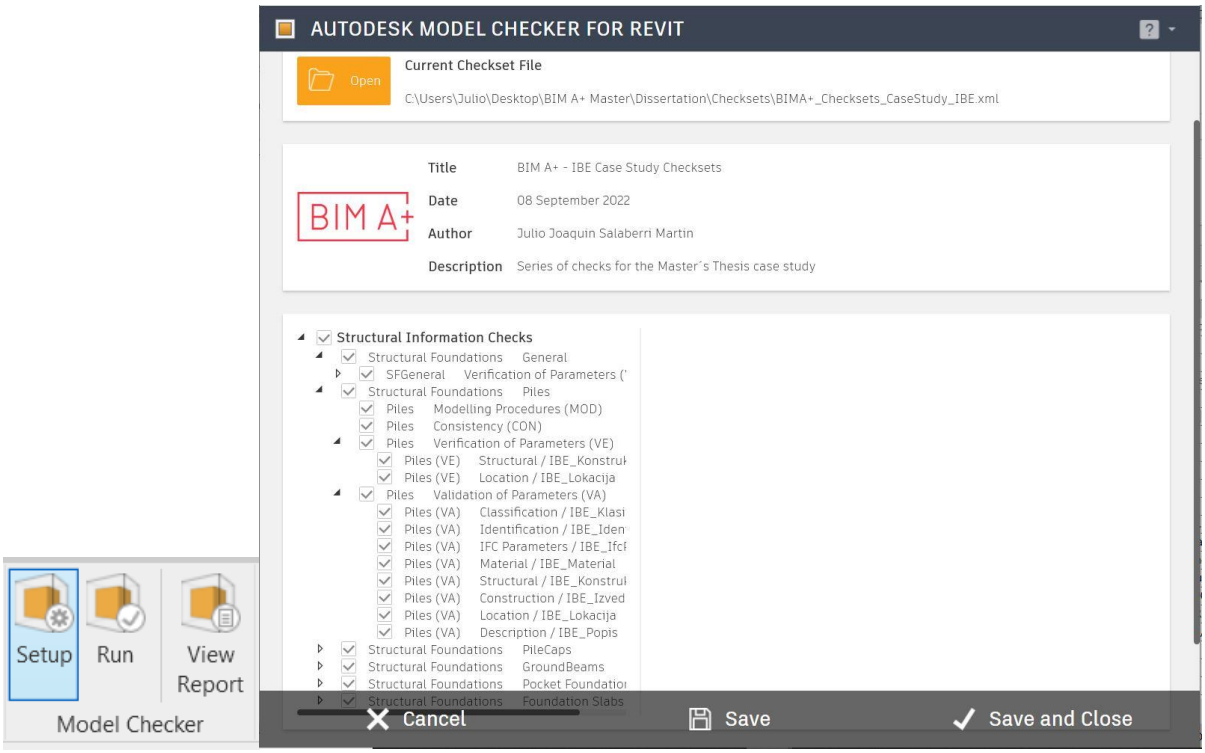

Figure 35: Check-set setup interface in Autodesk Model Checker for Revit

Because the project specific check-set was structured in a way that allows modularity and with a high level of granularity, it is possible to select exactly which checks it is desired to perform in the model. This is very useful, for example, when one member of the project team might be working on some of the elements and wishes to perform an isolated automated check to evaluate the current state of the elements and perform active corrective measures over the elements, instead of waiting for a planned complete check of the model to be performed by the BIM Coordinator.

Once the check-set is associated with the model, it can be run by clicking the "Run" button. A new interface will appear that will allow to run the check-set under the button "Run Report". The tool offers the possibility of adding other BIM models to be checked, not only the current one, and the possibility

of also checking the models that are linked to the current model. This feature can be useful if, for example, the design company has a "Master Model" in the Authoring Tool for the current project where all the discipline models are linked.

Once the "Run Report" button is clicked, the tool will perform the checks that were selected to be performed under the setup option mentioned before.

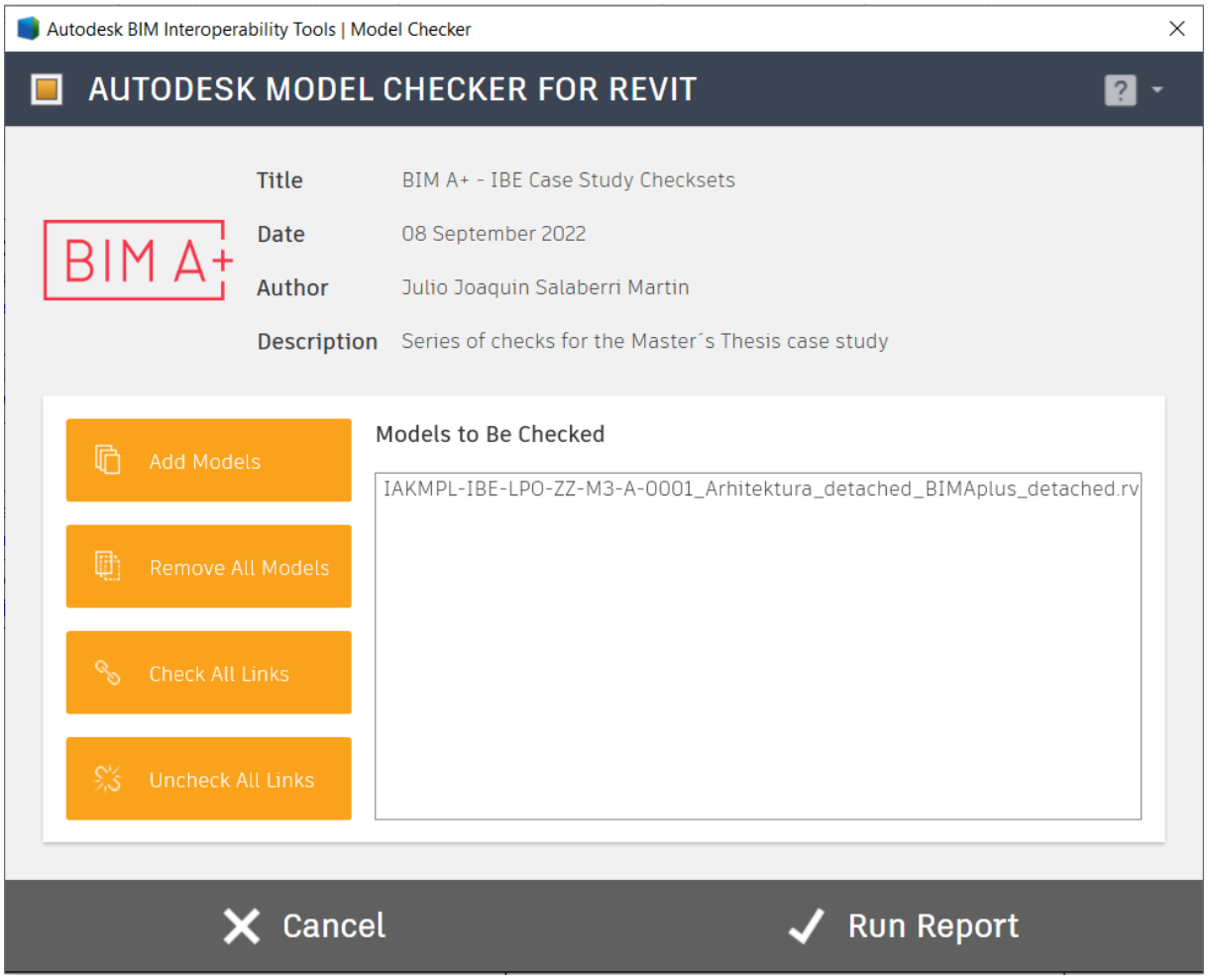

Figure 36: Interface of Model Checker for Revit to run the desired check-set

Aside from this "manual" way to perform the automated checks in the desired model, there is also the possibility to automate this process up to a certain level with this tool. This allows this process to be "hands-off" as much as possible. However, this possibility has not been explored for this thesis.

4.6 Methodology application: Step 4 – Rule Check Reporting

Once the checks are performed after selecting the "Run Report" option, the Autodesk Model Checker for Revit tool will show an interface with the report of the results of the checks. The results reported depend in a direct way on how the project specific rule-based checks were built. In the present case study, the tool will report all the instances that do not pass each check, and a check fails when one or more instances do not pass the rule-based check. The tool also shows a percentage, that indicates the percentage of checks that were passed over the total number of checks performed. It is worth mentioning that this percentage is not, in any way, a quantitative measure of the quality of the model. However, some companies that use this tool for automated model-checking associate the percentage of the checks passed with a level of quality of the model, and they consider that the model is of quality when a certain percentage of passed checks is reached.

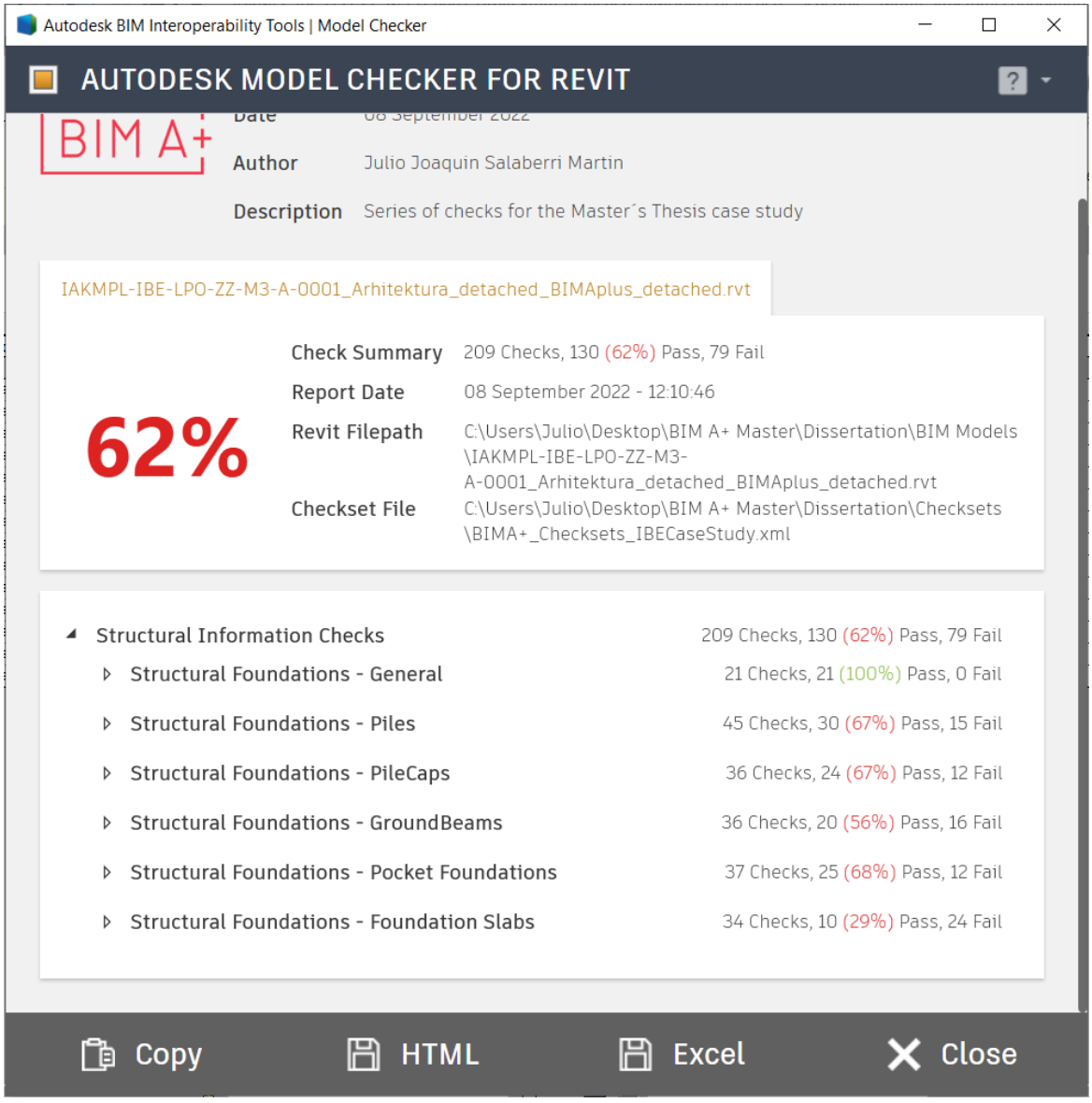

Figure 37: Interface of Model Checker for Revit with the results of the run check-set

The report generated by the tool allows to expand the results on the check-set structure to reach the desired check and consult which instances are not complying with the defined requirements in the checks. This option enhances the first type of reporting identified in the methodology: **Information issues reporting.**

The issues found can be reported to a team member using BCF Format with a tool like BCF Manager for Revit. This is illustrated in the following images.

First, the elements can be consulted in the Model Checker interface, and by clicking the lens icon they can be selected in the model:

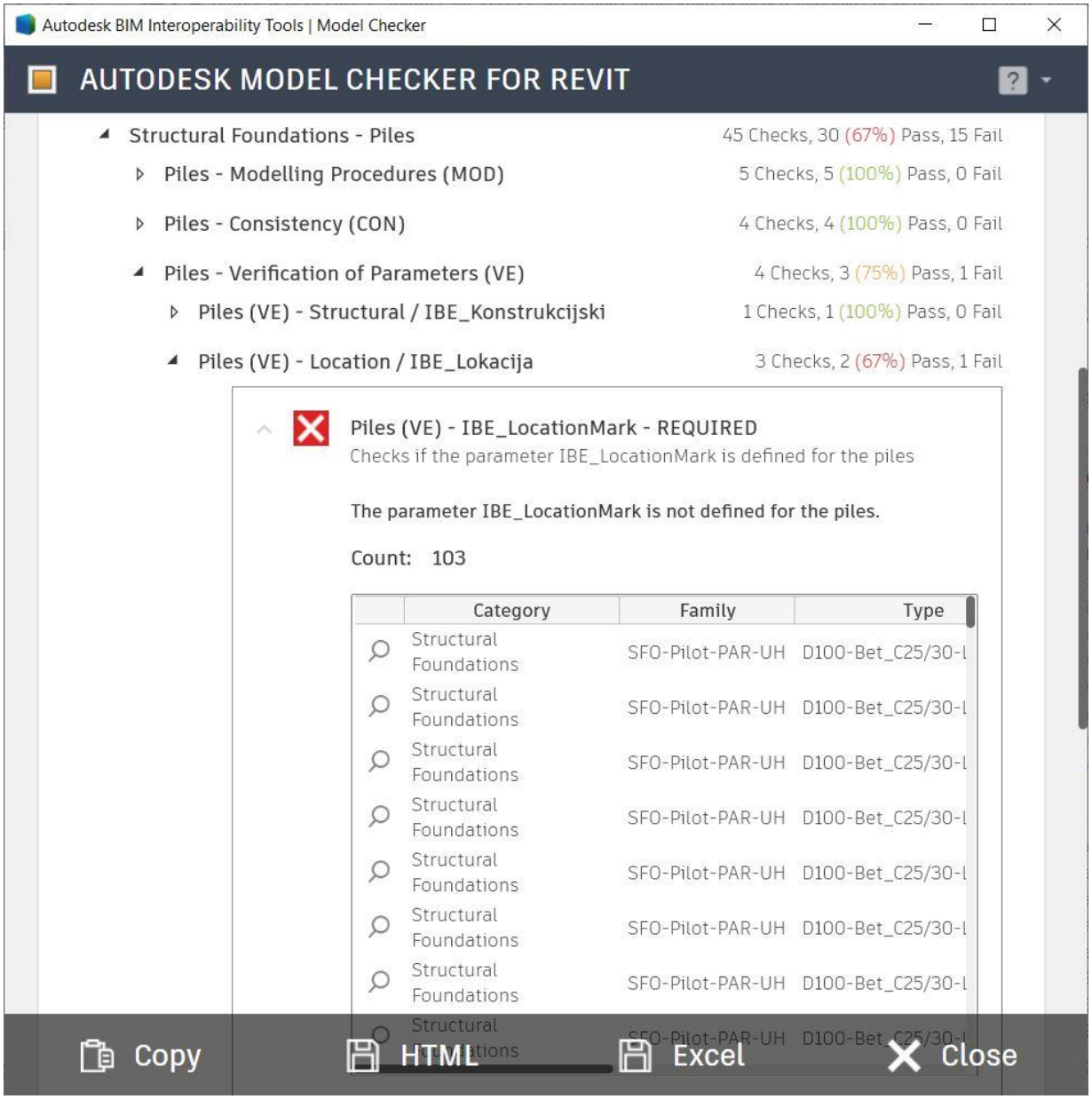

Figure 38: Model Checker for Revit interface with the report of the results of the check-set

Secondly, a new issue can be created by a tool like BIMCollab BCF Manager to communicate to a design team member the results of the check and if necessary, ask to take corrective actions on the elements that have not pass the checks successfully.

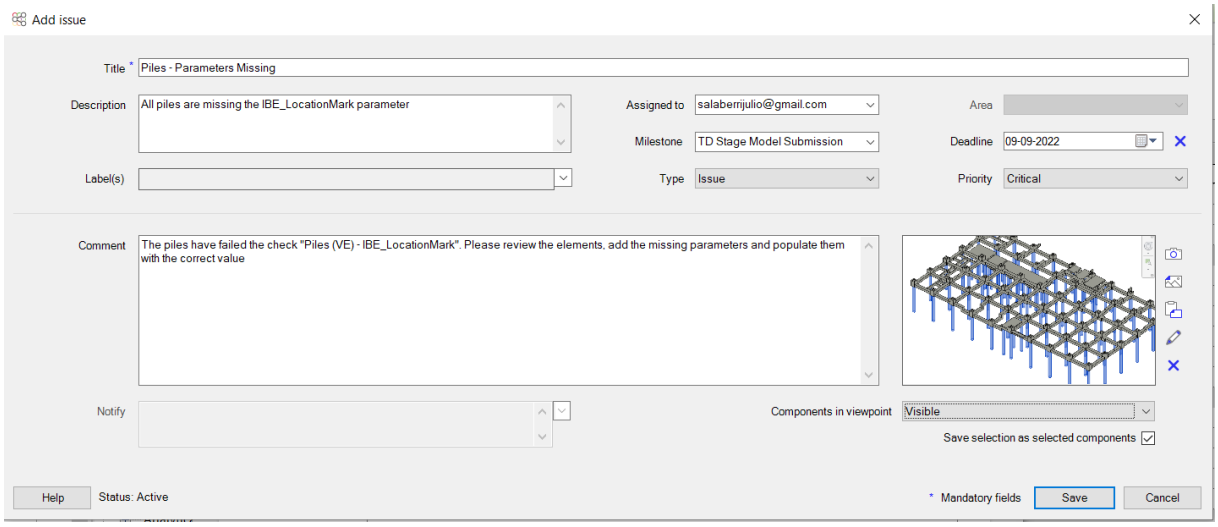

Figure 39: Reporting an issue to a team member for the elements that have not passed a certain check with BCF Manager from BIMCollab

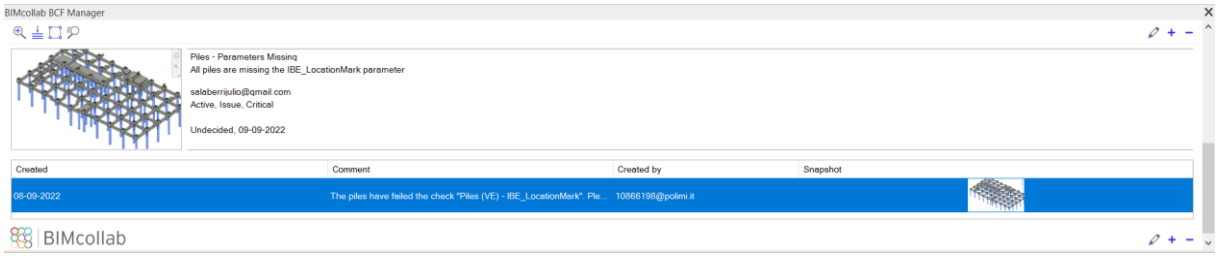

Figure 40: Issues report with BCF Manager tool in Revit

As can be seen in Figures 37 and 38, the tool also supports the possibility of exporting the result of the checks performed in excel format. This possibility enhances the second type of reporting mentioned in the methodology: **Management reporting.**

Once the results are exported, the data in the excel file can be organized and presented in a way that supports the team members that are responsible for management and decision making to evaluate the results and come to different conclusions based on the data presented.

For this kind of reporting, the use of Microsoft Power BI tool has been explored. This tool can be used to create a dashboard that reads the results of the checks from the excel file created and present them in a graphical way that allows to understand easily what it is desired to transmit.

Figure 41: Management report in Power BI for the results of the automated checks

In addition, it is also possible to setup a dashboard in Power BI in a way that reads and compares the reports from different days located in the same folder. This can also be useful for project specific decision making.

5 CONCLUSION

The workflow proposed in this thesis aims to identify the steps needed to be taken to successfully implement automated rule-based checks with the objective of validating the alphanumerical information content of a BIM model, into a design company QA or QC plans and design phase processes. The objective of this thesis was not to try to implement quantitative measures of quality in the models, but to analyse how the model checking activities can be improved by defining a proper workflow for the implementation and the use of rule-based checking tools withing the authoring tool environment that are already available in the market.

By testing the workflow in the case study provided by the partner institution, IBE d.d., it has been possible to evaluate the strengths and limitations of the workflow, and if it confirms the hypothesis formulated in the methodology chapter: "*the implementation of automated checking procedures as part of the QA/QC processes in the authoring tool environment, can help save time that designers spend for that task, avoiding source of errors like exporting processes, and improve the efficiency of the information production and the control over such information, which, ultimately, translates into money saved by the client".*

Under the point of view of the author, the application of the workflow on the case study has proven the hypothesis be right, but under certain circumstances, this might not be the case.

For the hypothesis to be corroborated, the different steps of the proposed workflow need to be implemented into the design company processes and across different projects, and not being implemented as an isolated process. This is because the effort to carry out the steps of "Rule Interpretation and "Building Model Preparation" the first time is tangible. If the check-sets are well structured and the rule-sets are built in a way that enhance replicability, modularity and ability to be stored, the effort needed to create project specific check-sets gets drastically reduced, being the activity that needs more time to be carried out the "Building Model Preparation".

The testing of the workflow on the case study has corroborated that after a certain number of checks have been carried out with automated rule-based checks, the time spent in implementing the checks gets compensated, even more if the automated rule-base checks are implemented the earliest as possible during the design stages. Estimating that a total of 20 hours is needed to implement the automated rule-based checks in a project for the first time in the company, and these checks need 3 hours of the time of a person to be performed in a manual way, after the 7th time that the automated checks are performed, they start to save time for the design company, and therefore, money. If the implemented checks in the first project can later be replicated for another project and just 10 hours are needed to implement them, the company will start saving money just after the 3rd time that the checks are performed (all this just

assuming that the checks are only performed before a deadline or a milestone, and not for self-assessment or modelling support).

The rule-base checks for the case study have been built in a way that they could be replicated between the different type of elements that was desired to be checked. While creating the checks for the piles took a certain amount of time (it should be mentioned that the author had no experience with the model checking tool selected for the case study, for what there was a learning curve), the amount of time needed to create the same checks for the pile caps just took a quarter of that time, and the same for the other types of elements. This was also enhanced by the possibility of editing the rule-based checks in a source-code editor.

It has also been possible to verify that in order to replicate and implement the rule-based checks developed in other projects in a successful way, the design company should have very strong modelling and naming standards. This is because, much of the alphanumerical information contained in the model is the type of text, and the checking can be carried out using Regular Expression, and a change on the structure on how the information is presented will enforce that the rule-based checks will need to be edited to value the object´s properties accurately.

The implementation of the workflow has also been identified to have certain limitations which are directly related to the software tools chosen. To illustrate this, both the authoring software and the model checking tool used for the case study do not treat the same way the system families of the project and the loadable families. This can be a limitation on the way that the rule-based checks are built, specifically the part of the rules that intend to identify the object to be evaluated. Another limitation identified in the model checking tool has been the impossibility of comparing two properties' values, which could be useful, for example, to check the consistency between the name of an object and the information available in other of the object parameters.

In general, the case study has proven that the implementation of automated checks for validating the alphanumerical information of the objects against given requirements in the environment of the Authoring Tool is possible with the available technology, some manual or semi-automated model checking procedures can be substituted by automated rule-based checks, and it does not necessarily need to be carried out using IFC format. This also saves a considerable amount of time since it is no longer necessary to export the model to IFC to validate the content afterwards.

Some possible lines of works have been identified from the developing of this thesis. In the literature review, validation of the model has been mentioned and the implementation of automated rule-based checks have been proposed in this thesis to carry out that validation, but the topics of clash detection and code checking within the environment of Authoring Tool can also be investigated. In addition, the

automated performing of the rule-based checks, this means the elimination of the necessity of a human being triggering the process of the checking could also be a potential topic to be researched about.

6 REFERENCES

[1] The International Organization for Standardization (ISO), 'ISO 19650-1:2018 Organization of information about construction works - Information management using building information modelling - Part 1: Concepts and Principles.' 2018.

[2] C. Mirarchi and A. Pavan, 'Building information models are dirty', in *Proceedings of the 2019 European Conference on Computing in Construction*, Jul. 2019, pp. 131–140. doi: 10.35490/EC3.2019.180.

[3] H. Kulusjärvi, 'COBIM 2012 Series 6: Quality Assurance'. Mar. 2012. Accessed: Aug. 01, 2022. [Online]. Available: https://buildingsmart.fi/wp-content/uploads/2016/11/cobim_6_quality_assurance_v1.pdf

[4] Oxford Advanced Learner's Dictionary, 'Quality'. https://www.oxfordlearnersdictionaries.com/definition/english/quality_1?q=quality (accessed Jun. 01, 2022).

[5] BIM Dictionary, 'Quality Definiton'. https://bimdictionary.com/en/quality/1 (accessed May 26, 2022).

[6] B. Succar, 'Building information modelling framework: A research and delivery foundation for industry stakeholders', *Automation in Construction*, vol. 18, no. 3, pp. 357–375, May 2009, doi: 10.1016/j.autcon.2008.10.003.

[7] BIM Dictionary, 'Digital Asset Definition'. https://bimdictionary.com/en/digital-asset/1 (accessed Jul. 11, 2022).

[8] V. Donato, M. Lo Turco, and M. M. Bocconcino, 'BIM-QA/QC in the architectural design process', *Architectural Engineering and Design Management*, vol. 14, no. 3, pp. 239–254, May 2018, doi: 10.1080/17452007.2017.1370995.

[9] C. M. Eastman, Ed., *BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors*, 2nd ed. Hoboken, NJ: Wiley, 2011.

[10] I. Robledo, A. Mendoza, and R. Perea, 'Datos vs. Información'. http://www3.uacj.mx/CGTI/CDTE/JPM/Documents/IIT/Introduccion_TI/3_Modelos_sistemas/datos-vs.-informaci%c3%b3n.html (accessed Jul. 13, 2022).

[11] R. Y. Wang and D. M. Strong, 'Beyond Accuracy: What Data Quality Means to Data Consumers', *Journal of Management Information Systems*, vol. 12, no. 4, pp. 5–33, 1996.

[12] F. Naumann and C. Rolker, *Assessment Methods for Information Quality Criteria*. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Informatik, 2005. doi: 10.18452/2441.

[13] W. Delone and E. McLean, 'The DeLone and McLean Model of Information Systems Success: A Ten-Year Update', *J. of Management Information Systems*, vol. 19, pp. 9–30, Apr. 2003, doi: 10.1080/07421222.2003.11045748.

[14] W. Solihin, C. Eastman, and Y.-C. Lee, 'Toward robust and quantifiable automated IFC quality validation', *Advanced Engineering Informatics*, vol. 29, no. 3, pp. 739–756, Aug. 2015, doi: 10.1016/j.aei.2015.07.006.

[15] 'AIA'. https://www.aia.org/ (accessed Jul. 26, 2022).

[16] 'Level of Development Specification – BIM Forum'. https://bimforum.org/lod/ (accessed Jul. 26, 2022).

[17] 'DiKon BIM7AA Specification of Building Parts'. http://www.bim7aa.dk/DIKON_BIM7AA_Bygningsdelsspecifikationer_UK.html (accessed Jul. 27, 2022).

[18] A. Pavan, 'Sistema dei LOD italiano: UNI 11337-4 2017', *Ingenio Informazione tecnica e progettuale*. https://www.ingenio-web.it/18667-sistema-dei-lod-italiano-uni-11337-4-2017 (accessed Jul. 26, 2022).

[19] European Committee for Standardization, 'EN 17412-1:2021 Building Information Modelling - Level of Information Need - Part 1: Concepts and principles'.

[20] 'BS EN 17412-1:2020 Building information modelling - level of information need. Concepts and principles, British Standards Institution - Publication Index | NBS'. https://www.thenbs.com/PublicationIndex/documents/details?Pub=BSI&DocID=330918 (accessed Jul. 27, 2022).

[21] '211in Model Uses List', *BIMe Initiative*. https://bimexcellence.org/resources/200series/211in/ (accessed Jul. 27, 2022).

[22] 'RIBA Plan of Work'. https://www.architecture.com/knowledge-and-resources/resources-landing-page/riba-plan-of-work (accessed Jul. 27, 2022).

[23] BIM Dictionary, 'Quality Assurance Definition'. https://bimdictionary.com/en/quality-assurance/1 (accessed Jul. 30, 2022).

[24] BIM Dictionary, 'Quality Control Definition'. https://bimdictionary.com/en/quality-control/1 (accessed Jul. 30, 2022).

[25] BIM Dictionary, 'Quality Management Definition'. https://bimdictionary.com/en/quality-management/1 (accessed Jul. 30, 2022).

[26] 'The Difference Between Quality Assurance and Quality Control - Open Dialog - Dialog Information Technology'. https://www.dialog.com.au/open-dialog/the-difference-between-quality-assurance-and-quality-control/ (accessed Jul. 30, 2022).

[27] Singapore Building and Construction Authority, 'Singapore BIM Guide Version 2.0'. Accessed: Jul. 30, 2022. [Online]. Available: https://www.corenet.gov.sg/media/586132/Singapore-BIM-Guide_V2.pdf

[28] Puertos del Estado, 'Guía BIM del Sistema Portuario de Titularidad Estatal'. Jun. 2019.

[29] Euskal Trembide Sarrea, 'Manual BIM de ETS'. Apr. 2020. Accessed: Jul. 31, 2022. [Online]. Available: https://www.ets-rfv.euskadi.eus/contenidos/informacion/ets_bim/es_def/adjuntos/MANUAL_BIM_ETS.pdf

[30] Junta de Extremadura, 'Guía BIM de la Dirección General de Movilidad e Infraestructuras viarias'. Jun. 2021. Accessed: Jul. 31, 2022. [Online]. Available: http://www.juntaex.es/filescms/con07/uploaded_files/dgi/Guia_BIM_DG_Movilidad_Junta_Extremadura.pdf

[31] Ferrocarrils de la Generalitat Valenciana, 'Manual BIM FGV'. Oct. 2020. Accessed: Jul. 31, 2022. [Online]. Available: https://www.fgv.es/manual_bim

[32] Building Smart Spanish Chapter, 'Guia de usuarios BIM Documento 6: Aseguramiento de la Calidad'. Oct. 2014.

[33] T. Barichello Bohrer, 'Quality assurance of BIM models in project management', University of Ljubljana, 2021. Accessed: Apr. 19, 2022. [Online]. Available: https://repozitorij.uni-lj.si/IzpisGradiva.php?id=131602&lang=slv&prip=rul:11935545:r3

[34] 'buildingSMART - The International Home of BIM', *buildingSMART International*. https://www.buildingsmart.org/ (accessed Jul. 31, 2022).

[35] A. T. Kovacs and A. Micsik, 'BIM quality control based on requirement linked data', *International Journal of Architectural Computing*, vol. 19, no. 3, pp. 431–448, Sep. 2021, doi: 10.1177/14780771211012175.

[36] D. D. Vendrell, 'IFC en entorno de autoría BIM: buenas prácticas para exportar e importar', *BIM European Summit Barcelona*, p. 47, 2020.

[37] D. D. Vendrell, 'Gestión de modelos IFC: auditoría y visualización', *BIM European Summit Barcelona*, p. 47, 2020.

[38] BIM Dictionary, 'Model Validation'. https://bimdictionary.com/en/model-validation/1 (accessed Jul. 31, 2022).

[39] A. Ciribini, S. Mastrolembo Ventura, and M. Bolpagni, 'Informative content validation is the key to success in a BIM-based project', *Territorio Italia - Governo del Territorio, Catasto, Mercato immobiliare*, vol. 2–2015, pp. 9–30, Jan. 2016, doi: 10.14609/Ti_2_15_1e.

[40] E. Hjelseth, 'BIM-based model checking (BMC)', in *Building Information Modeling: Applications and Practices*, 2015, pp. 33–61. doi: 10.1061/9780784413982.ch02.

[41] E. Hjelseth, 'Classification of BIM-based Model checking concepts', *Journal of Information Technology in Construction (ITcon)*, vol. 21, pp. 354–369, Oct. 2016.

[42] Autodesk, 'Generative design for architecture, engineering & construction | Autodesk'. https://www.autodesk.com/solutions/generative-design/architecture-engineering-construction (accessed Aug. 04, 2022).

[43] J. Ocean, 'BIM Clash Detection: Definition, Benefits and Software Types', *Revizto*, Aug. 18, 2021. https://revizto.com/en/clash-detection-in-bim/ (accessed Aug. 03, 2022).

[44] S. Mastrolembo Ventura, V. Getuli, P. Capone, and A. Ciribini, *BIM-based Code Checking for Construction Health and Safety*. 2017. doi: 10.1016/j.proeng.2017.07.224.

[45] R. A. Niemeijer, de Vries B., and J. Beetz, 'Check-mate: Automatic constraint checking of IFC models: 26th International CIB W78 Conference, October 1-3, 2009, Istanbul, Turkey', *Managing IT in Construction/Managing Construction for Tomorrow*, pp. 479–486, 2009.

[46] C. Eastman, J. Lee, Y. Jeong, and J. Lee, 'Automatic rule-based checking of building designs', *Automation in Construction*, vol. 18, no. 8, pp. 1011–1033, Dec. 2009, doi: 10.1016/j.autcon.2009.07.002.

[47] A. S. Ismail, K. N. Ali, and N. A. Iahad, 'A Review on BIM-based automated code compliance checking system', in *2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*, Jul. 2017, pp. 1–6. doi: 10.1109/ICRIIS.2017.8002486.

[48] 'Solibri Office - The core product for model checking and collaboration', *Solibri*. https://www.solibri.com/solibri-office (accessed Apr. 19, 2022).

[49] 'CORENET e-PlanCheck: Singapore's Automated Code Checking System-- AECbytes Article'. https://www.aecbytes.com/feature/2005/CORENETePlanCheck.html (accessed Aug. 05, 2022).

[50] E. Hjelseth and N. Nisbet, 'Capturing normative constraints by use of the semantic mark-up RASE methodology', in *Proceedings of the 28th International Conference of CIB W78*, Sophia Antipolis, France., 2011, pp. 1–10.

[51] W. Solihin and C. Eastman, 'Classification of rules for automated BIM rule checking development', *Automation in Construction*, vol. 53, pp. 69–82, May 2015, doi: 10.1016/j.autcon.2015.03.003.

[52] 'Construction Management Software | Autodesk Construction Cloud'. https://construction.autodesk.com/ (accessed Aug. 23, 2022).

[53] 'BIMcloud', *Graphisoft*. https://graphisoft.com/solutions/bimcloud (accessed Aug. 23, 2022).

[54] 'Autodesk BIM Interoperability Tools Master Page'. https://interoperability.autodesk.com/ (accessed Aug. 27, 2022).

[55] 'BIM Collaboration Format (BCF)', *buildingSMART International*. https://www.buildingsmart.org/standards/bsi-standards/bim-collaboration-format-bcf/ (accessed Aug. 27, 2022).

[56] 'Autodesk Model Checker Configurator'. https://interoperability.autodesk.com/modelcheckerconfigurator.php (accessed Sep. 06, 2022).

[57] 'Autodesk Model Checker for Revit'. https://interoperability.autodesk.com/modelchecker.php (accessed Sep. 06, 2022).

[58] 'About | The Dynamo Primer'. https://primer.dynamobim.org/ (accessed Sep. 07, 2022).

[59] 'Power BI Desktop—Interactive Reports | Microsoft Power BI'. https://powerbi.microsoft.com/en-us/desktop/ (accessed Sep. 07, 2022).

7 ANNEXES

7.1 ANNEX 1: ALPHANUMERICAL INFORMATION REQUIREMENTS

Alphanumerical Requirements - Piles		
Revit Parameter	Type of Parameter	Example of Value
IBE_ElementType	Text	Pilot - uvrtan
Mark	Text	PIL-K1-001
Description	Text	Uvrtan pilot D=100 cm, L=25 m
Comments	Text	<komentar>
Type Comments	Text	PIL-D100-A
Type Mark	Text	P-01
IfcExportAs	Text	IfcPile
IfcExportType	Text	BORED
Diameter	Length	1 m
Length	Length	25 m
Elevation at Top	Length	-8 m
Elevation at Bottom	Length	-30 m
Structural Material	Text	Beton-C25/30
IBE_ConcreteGrade	Text	C25/30
IBE_ExposureClass	Text	XC0
IBE_ChlorideClass	Text	Cl 0.2
IBE_ConcreteDmax	Text	D16
IBE_ConcreteSlumpClass	Text	S3
Structural	Boolean	Yes
IBE_ReinforcementGrade	Text	B500B
IBE_ReinforcementRatio	Integer	80
IBE_RebarCover	Length	0.04
IBE_PileBearingCapacity	Force	7,0 kN
IBE_ConstructionProcess	Text	InSitu
IBE_ExecutionClass	Text	EXC2
IBE_SurfaceTreatment	Text	VB0
IBE_CuringClass	Text	1
IBE_ToleranceClass	Text	1
IBE_SideFormworkType	Text	JeklenaCev
Level	Text	K01
IBE_LocationMark	Text	A(+2,6)1
IBE_CoordinateX	Length	15 m
IBE_CoordinateY	Length	15 m
Keynote	Text	01.07.01.XXX
IBE_OpisPostavkeIZS	Text	Uvrtan pilot D=100 cm, C25/30
IBE_Volume	Volume	19,6 m3
IBE_PileShaftArea	Area	78,5 m2
IBE_RebarQuantityEstimate	Real	78,5 m2
IBE_BOQReference	Text	1.3.25

Alphanumerical Requirements - Pile Caps		
Revit Parameter	Type of Parameter	Example of Value
IBE_ElementType	Text	Pilotna greda - pravokotna
Mark	Text	HBM-K1-001
Description	Text	Pilotna greda B/H=160x60 cm
Comments	Text	<komentar>
Type Comments	Text	HBM-Tip A
Type Mark	Text	HBM-01
IfcExportAs	Text	IfcFooting
IfcExportType	Text	FOOTING_BEAM
Thickness	Length	0,6 m
Width	Length	1,6 m
Length	Length	33,80 m
Elevation at Top	Length	-8 m
Elevation at Bottom	Length	-8,6 m
Structural Material	Text	Beton-C30/37
IBE_ConcreteGrade	Text	C30/37
IBE_ExposureClass	Text	XC0
IBE_ChlorideClass	Text	Cl 0.2
IBE_ConcreteDmax	Text	D16
IBE_ConcreteSlumpClass	Text	S3
Structural	Boolean	Yes
IBE_ReinforcementGrade	Text	B500B
IBE_ReinforcementRatio	Integer	80
IBE_RebarCover	Length	0,04
IBE_ConstructionProcess	Text	InSitu
IBE_ExecutionClass	Text	EXC2
IBE_SurfaceTreatment	Text	VB0
IBE_CuringClass	Text	1
IBE_ToleranceClass	Text	1
IBE_SideFormworkType	Text	Stranski opaž pasovnih temeljev
Level	Text	K01
Keynote	Text	01.07.01.XXX
IBE_OpisPostavkeIZS	Text	Pilotna greda B/H=120x40 cm, C30/37
Thickness	Length	0,6 m
Volume	Volume	32,45 m3
IBE_NetSideFormworkArea	Area	48,48 m2
IBE_BOQReference	Text	1.3.25
IBE_RebarQuantityEstimate	Real	78,5 m2

Alphanumerical Requirements - Ground Beams		
Revit Parameter	Type of Parameter	Example of Value
IBE_ElementType	Text	Pasovni temelj
Mark	Text	STF-K01-001
Description	Text	Pasovni temelj B/H=100/50 cm
Comments	Text	<komentar>
Type Comments	Text	STF-Tip 1
Type Mark	Text	STF-01
IfcExportAs	Text	IfcFooting
IfcExportType	Text	STRIP_FOOTING
Length	Length	10,0 m
Width	Length	1,0 m
Thickness	Length	0,5 m
Elevation at Top	Length	-8,0 m
Elevation at Bottom	Length	-8,5 m
Structural Material	Text	Beton-C30/37
IBE_ConcreteGrade	Text	C30/37
IBE_ExposureClass	Text	XC2
IBE_ChlorideClass	Text	Cl 0.2
IBE_ConcreteDmax	Text	D16
IBE_ConcreteSlumpClass	Text	S3
IBE_ReinforcementGrade	Text	B500B
IBE_ReinforcementRatio	Integer	100
IBE_RebarCover	Length	0,04
IBE_ConstructionProcess	Text	InSitu
IBE_ExecutionClass	Text	EXC2
IBE_SurfaceTreatment	Text	VB0
IBE_CuringClass	Text	1
IBE_ToleranceClass	Text	1
IBE_SideFormworkType	Text	Opaz pasovnih temeljev
Level	Text	K01
Keynote	Text	01.07.01.05G
IBE_OpisPostavkeIZS	Text	Pasovni temelj B/H=50x100 cm, C30/37
Length	Length	10,0 m
Width	Length	1,0 m
Thickness	Length	0,5 m
Volume	Volume	5,0 m3
IBE_NetSideFormworkArea	Area	11,0 m2
IBE_BOQReference	Text	1.3.25
IBE_RebarQuantityEstimate	Real	78,5 m2

Alphanumerical Requirements - Pocket Foundations		
Revit Parameter	Type of Parameter	Example of Value
IBE_ElementType	Text	Temeljna casa
Mark	Text	PCF-K01-001
Description	Text	Temeljna casa L/B/H=180/180/250 cm
Comments	Text	<komentar>
Type Comments	Text	PCF-Tip 1
Type Mark	Text	PCF-01
IfcExportAs	Text	IfcFooting
IfcExportType	Text	PAD_FOOTING
Length	Length	1,8 m
Width	Length	1,8 m
Thickness	Length	2,5 m
Elevation at Top	Length	-8,0 m
Elevation at Bottom	Length	-10,5 m
Structural Material	Text	Beton-C30/37
IBE_ConcreteGrade	Text	C30/37
IBE_ExposureClass	Text	XC2
IBE_ChlorideClass	Text	Cl 0.2
IBE_ConcreteDmax	Text	D16
IBE_ConcreteSlumpClass	Text	S3
IBE_ReinforcementGrade	Text	B500B
IBE_ReinforcementRatio	Integer	100
IBE_RebarCover	Length	0,04
IBE_ConstructionProcess	Text	InSitu
IBE_ExecutionClass	Text	EXC2
IBE_SurfaceTreatment	Text	VB0
IBE_CuringClass	Text	1
IBE_ToleranceClass	Text	1
IBE_SideFormworkType	Text	Opaz temeljne case
Level	Text	K1
Keynote	Text	01.07.01.05X
IBE_OpisPostavkeIZS	Text	Temeljna casa L/B/H=180/180/250 cm, C30/37
Thickness	Length	2,5 m
Volume	Volume	6,6 m3
IBE_NetSideFormworkArea	Area	18 m2
IBE_BOQReference	Text	1.3.25
IBE_RebarQuantityEstimate	Real	78,5 m2

Alphanumerical Requirements - Foundation Slabs		
Revit Parameter	Type of Parameter	Example of Value
IBE_ElementType	Text	Temeljna plosca
Mark	Text	FSL-K01-001
Description	Text	Temeljna plosca t=80 cm
Comments	Text	<komentar>
Type Comments	Text	FSL-T80
Type Mark	Text	FSL-01
IfcExportAs	Text	IfcSlab
IfcExportType	Text	BASESLAB
Thickness	Length	0,8 m
Elevation at Top	Length	-7,2 m
Elevation at Bottom	Length	-8,0 m
Structural Material	Text	Beton-C30/37
IBE_ConcreteGrade	Text	C30/37
IBE_ExposureClass	Text	XC3
IBE_ChlorideClass	Text	Cl 0.2
IBE_ConcreteDmax	Text	D16
IBE_ConcreteSlumpClass	Text	S3
IBE_ReinforcementGrade	Text	B500B
IBE_ReinforcementRatio	Integer	100
IBE_RebarCover	Length	0,04
IBE_ConstructionProcess	Text	InSitu
IBE_ExecutionClass	Text	EXC2
IBE_SurfaceTreatment	Text	VB0
IBE_CuringClass	Text	1
IBE_ToleranceClass	Text	1
IBE_SideFormworkType	Text	Opaz roba temeljne plosce
Level	Text	K1
Keynote	Text	01.07.01.07X
IBE_OpisPostavkeIZS	Text	Temeljna plosca t=80 cm, C30/37
Thickness	Length	0,8 m
Lenght	Length	479,08 m
Area	Area	12880 m2
Volume	Volume	10304 m3
IBE_NetSideFormworkArea	Area	383,2 m2
IBE_BOQReference	Text	1.3.25
IBE_RebarQuantityEstimate	Real	78,5 m2

7.2 ANNEX 2: PROJECT SPECIFIC CHECK-SET

Check Name	Description	Failure Message
SFGeneral (VE) - IBE_ElementType	Checks if the parameter IBE_ElementType is defined for Structural Foundations	The parameter IBE_ElementType is not defined for Structural Foundations
SFGeneral (VE) - IfcExportAs	Checks if the parameter IfcExportAs is defined for Structural Foundations	The parameter IfcExportAs is not defined for Structural Foundations
SFGeneral (VE) - IfcExportType	Checks if the parameter IfcExportType is defined for Structural Foundations	The parameter IfcExportType is not defined for Structural Foundations
SFGeneral (VE) - IBE_ConcreteGrade	Checks if the parameter IBE_ConcreteGrade is defined for Structural Foundations	The parameter IBE_ConcreteGrade is not defined for Structural Foundations
SFGeneral (VE) - IBE_ExposureClass	Checks if the parameter IBE_ExposureClass is defined for Structural Foundations	The parameter IBE_ExposureClass is not defined for Structural Foundations
SFGeneral (VE) - IBE_ChlorideClass	Checks if the parameter IBE_ChlorideClass is defined for Structural Foundations	The parameter IBE_ChlorideClass is not defined for Structural Foundations
SFGeneral (VE) - IBE_ConcreteDmax	Checks if the parameter IBE_ConcreteDmax is defined for Structural Foundations	The parameter IBE_ConcreteDmax is not defined for Structural Foundations
SFGeneral (VE) - IBE_ConcreteSlumpClass	Checks if the parameter IBE_ConcreteSlumpClass is defined for Structural Foundations	The parameter IBE_ConcreteSlumpClass is not defined for Structural Foundations
SFGeneral (VE) - IBE_ReinforcementGrade	Checks if the parameter IBE_ReinforcementGrade is defined for Structural Foundations	The parameter IBE_ReinforcementGrade is not defined for Structural Foundations
SFGeneral (VE) - IBE_ReinforcementRatio	Checks if the parameter IBE_ReinforcementRatio is defined for Structural Foundations	The parameter IBE_ReinforcementRatio is not defined for Structural Foundations
SFGeneral (VE) - IBE_RebarCover	Checks if the parameter IBE_RebarCover is defined for Structural Foundations	The parameter IBE_RebarCover is not defined for Structural Foundations
SFGeneral (VE) - IBE_ConstructionProcess	Checks if the parameter IBE_ConstructionProcess is defined for Structural Foundations	The parameter IBE_ConstructionProcess is not defined for Structural Foundations
SFGeneral (VE) - IBE_ExecutionClass	Checks if the parameter IBE_ExecutionClass is defined for Structural Foundations	The parameter IBE_ExecutionClass is not defined for Structural Foundations
SFGeneral (VE) - IBE_SurfaceTreatment	Checks if the parameter IBE_SurfaceTreatment is defined for Structural Foundations	The parameter IBE_SurfaceTreatment is not defined for Structural Foundations
SFGeneral (VE) - IBE_CuringClass	Checks if the parameter IBE_CuringClass is defined for Structural Foundations	The parameter IBE_CuringClass is not defined for Structural Foundations
SFGeneral (VE) - IBE_ToleranceClass	Checks if the parameter IBE_ToleranceClass is defined for Structural Foundations	The parameter IBE_ToleranceClass is not defined for Structural Foundations
SFGeneral (VE) - IBE_SideFormworkType	Checks if the parameter IBE_SideFormworkType is defined for Structural Foundations	The parameter IBE_SideFormworkType is not defined for Structural Foundations
SFGeneral (VE) - IBE_OpisPostavkeIZS	Checks if the parameter IBE_OpisPostavkeIZS is defined for Structural Foundations	The parameter IBE_OpisPostavkeIZS is not defined for Structural Foundations
SFGeneral (VE) - IBE_RebarQuantityEstimate	Checks if the parameter IBE_RebarQuantityEstimate is defined for Structural Foundations	The parameter IBE_RebarQuantityEstimate is not defined for Structural Foundations
SFGeneral (VE) - IBE_BOQReference	Checks if the parameter IBE_BOQReference is defined for Structural Foundations	The parameter IBE_BOQReference is not defined for Structural Foundations
SFGeneral (VE) - IBE_NetSideFormworkArea	Checks if the parameter IBE_NetSideFormworkArea is defined for Structural Foundations	The parameter IBE_NetSideFormworkArea is not defined for Structural Foundations
Piles (MOD) - Hosted not WorkPlane-Based	Checks if the piles in the model are hosted. This is not desirable for Structural Foundations as the elements will not be associate to a Level.	There are piles families that are hosted, but their host is not a work plane. This is not correct. Please Review.
Piles (MOD) - Level	Checks if the piles in the model are attached to the correct level.	There are piles associated to a wrong level. Please Review.
Piles (MOD) - Elevation at Bottom	Checks if there are piles with wrong elevation at bottom.	There are piles that go deeper than designed. Please Review
Piles (MOD) - Elevation at Top	Checks if there are piles with wrong elevation at top.	There are piles with elevation at top above the ground level. Please Review
Piles (MOD) - Structural Workset	Checks if there are piles outside the Structural Worksets	There are piles that are not in the correct workset
Piles (CON) - Naming Convention	Checks if the piles naming follow the naming convention. This check reports instances.	There are pile instances that do not follow the naming convention. Please check that the following naming convention is followed: D(2or3digits)-Bet_C25/30-L=(1or2digits),(2digits)m. Examples: D90-Bet_C25/30-L=8,50m or D120-Bet_C25/30-L=13,10m.
Piles (CON) - Piles outside of defined checks	Checks if there are new pile types that are not included in the following checks: Piles - TypeName and Diameter, Piles - TypeName and Length. This check reports instances.	There are new piles types that cannot be checked in the next checks. The diameter code in the type name does not match D100 or D120, or their length code does not match L=9,50m, L=9,90m, L=11,05m, L=11,60m, L=12,40m, L=12,60m, L=12,80m, L=13,10m. Please review the elements and, if correct, add them to this rule and the following rules: Piles - TypeName and Diameter, Piles - TypeName and Length.
Piles (CON) - TypeName and Diameter	Checks if the piles diameter matches the one described in the TypeName or there are new pile types that do not fit the check rule and cannot be validated. This check reports instances.	There are piles where the diameter and the TypeName are not consistent, or there are new pile types that can`t be validated with this check, because their TypeName code is different than D100 or D120, or their diameter is different than 1m or 1,2m. Please, check the reported piles and, if they are correct, add them to the rule.

Piles (CON) - TypeName and Length	Checks if the piles length matches the one described in the TypeName or there are new pile types that do not fit the check rule and cannot be validated. This check reports instances.	There are piles where the Length and the TypeName are not consistent, or there are new pile types that can`t be validated with this check, because their TypeName code is different than L=9,50m, L=9,90m, L=11,05m, L=11,60m, L=12,40m, L=12,60m, L=12,80m, L=13,10m., or their Length is different than 9,50m, 9,90m, 11,05m, 11,60m, 12,40m, 12,60m, 12,80m, 13,10m.. Please, check the reported piles and, if they are correct, add them to the rule.
Piles (VE) - IBE_PileBearingCapacity	Checks if the parameter IBE_PileBearingCapacity is defined for the piles	The parameter IBE_PileBearingCapacity is not defined for the piles.
Piles (VE) - IBE_LocationMark	Checks if the parameter IBE_LocationMark is defined for the piles	The parameter IBE_LocationMark is not defined for the piles.
Piles (VE) - IBE_CoordinateX	Checks if the parameter IBE_CoordinateX is defined for the piles	The parameter IBE_CoordinateX is not defined for the piles.
Piles (VE) - IBE_CoordinateY	Checks if the parameter IBE_CoordinateY is defined for the piles	The parameter IBE_CoordinateY is not defined for the piles.
Piles (VA) - IBE_ElementType	Checks if the parameter IBE_ElementType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ElementType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "Pilot".
Piles (VA) - OmniClass Number	Checks if the built-in parameter "OmniClass Number" has a correct value. This check reports instances.	The built-in parameter "OmniClass Number" does not have a correct value. The value should be "23.25.05.11"
Piles (VA) - OmniClass Title	Checks if the built-in parameter "OmniClass Title" has a correct value. This check reports instances.	The built-in parameter "OmniClass Title" does not have a correct value. The value should be "Foundation Piles"
Piles (VA) - Mark	Checks if the built-in parameter "Mark" has a correct value. This check reports instances.	The built-in parameter "Mark" does not have a correct value. The value should follow the format "PIL-K01-001".
Piles (VA) - Description	Checks if the built-in parameter "Description" has a correct value. This check reports instances.	The built-in parameter "Description" does not have a correct value. The value should follow the format "Uvrtan pilot D=100 cm, L=9,50 m". This check only validates the values D=100, D=120 and L=9,50 m, L=9,90 m, L=11,05 m, L=11,60 m, L=12,40 m, L=12,60 m, L=12,80 m, L=13,10 m. Please review the elements and, if correct, add them to this rule.
Piles (VA) - Type Comments	Checks if the built-in parameter "Type Comments" has a correct value. This check reports instances.	The built-in parameter "Type Comments" does not have a correct value. The value should follow the format "PIL-D100-A". This check only validates the values D100, D120. Please review the elements and, if correct, add them to this rule.
Piles (VA) - Type Mark	Checks if the built-in parameter "Type Mark" has a correct value. This check reports instances.	The built-in parameter "Type Mark" does not have a correct value. The value should follow the format "P-01". This check only validates the values from P-01 to P-13. Please review the elements and, if correct, add them to this rule.
Piles (VA) - IfcExportAs	Checks if the parameter IfcExportAs is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportAs is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "IfcPile".
Piles (VA) - IfcExportType	Checks if the parameter IfcExportType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "BORED".
Piles (VA) - Structural Material	Checks if the built-in parameter "Structural Material" has a correct value. This check reports instances.	The built-in parameter "Structural Material" does not have a correct value. The name of the Structural Material should begin with: IBE-Beton-C25/30
Piles (VA) - IBE_ConcreteGrade	Checks if the parameter IBE_ConcreteGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteGrade is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "C25/30". If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_ExposureClass	Checks if the parameter IBE_ExposureClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExposureClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "XC0, XC2 or XC4". If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_ChlorideClass	Checks if the parameter IBE_ChlorideClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ChlorideClass is not defined for this category, or the value is wrong for the following elements. Accepted values for this parameter are: "Cl 0.00" up to "Cl 1.00" (this check also accepts 1 decimal number as valid: 0..2". If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_ConcreteDmax	Checks if the parameter IBE_ConcreteDmax is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteDmax is not defined for this category, or the value is wrong for the following elements. The ccepted values for this parameter are: D8, D16 or D32. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_ConcreteSlumpClass	Checks if the parameter IBE_ConcreteSlumpClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteSlumpClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: S1, S2, S3, S4 or S5.
Piles (VA) - IBE_ReinforcementGrade	Checks if the parameter IBE_ReinforcementGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementGrade is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: B500B. If it should be of a different value, please include it to the rule.

Piles (VA) - IBE_ReinforcementRatio	Checks if the parameter IBE_ReinforcementRatio is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementRatio is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0-300 kg/m3. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_RebarCover	Checks if the parameter IBE_RebarCover is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarCover is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0,04m. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_PileBearingCapacity	Checks if the parameter IBE_PileBearingCapacity is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_PileBearingCapacity is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are >0 kN. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_ConstructionProcess	Checks if the parameter IBE_ConstructionProcess is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConstructionProcess is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: InSitu. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_ExecutionClass	Checks if the parameter IBE_ExecutionClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExecutionClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: EXC2. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_SurfaceTreatment	Checks if the parameter IBE_SurfaceTreatment is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SurfaceTreatment is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: VB0. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_CuringClass	Checks if the parameter IBE_CuringClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_CuringClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_ToleranceClass	Checks if the parameter IIBE_ToleranceClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ToleranceClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_SideFormworkType	Checks if the parameter IBE_SideFormworkType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SideFormworkType is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: JeklenaCev. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_LocationMark	Checks if the parameter IBE_LocationMark is defined and if it is, if the value of the parameter is correct. This parameter is used to indicate the location of the piles according to the grids (A-F, 11-20) + Symbol means right/up the grid. This check reports instances.	The parameter IBE_LocationMark is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "A-F(+-5,0)11-20(+-5,0). Example: "E(+2,0)15(0,0)" If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_CoordinateX	Checks if the parameter IBE_CoordinateX is defined and if it is, if the value of the parameter is correct. This parameter is used to indicate the absolute X coordinate of the piles in relation to the A-11 grids. This check reports instances.	The parameter IBE_CoordinateX is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0 to 50 (meters). If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_CoordinateY	Checks if the parameter IBE_CoordinateY is defined and if it is, if the value of the parameter is correct. This parameter is used to indicate the absolute X coordinate of the piles in relation to the A-11 grids. This check reports instances.	The parameter IBE_CoordinateY is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0 to 90 (meters). If it should be of a different value, please include it to the rule.
Piles (VA) - Keynote	Checks if the built-in parameter "Keynote" has a correct value. This check reports instances.	The built-in parameter "Keynote" does not have a correct value. The value should follow the format "01.07.01.XXX".
Piles (VA) - IBE_OpisPostavkeIZS	Checks if the parameter IBE_OpisPostavkeIZS is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_OpisPostavkeIZS is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "Uvrtan pilot D=(100 or 120) cm, C25/30". If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_RebarQuantityEstimate	Checks if the parameter IBE_RebarQuantityEstimate is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarQuantityEstimate is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: >=0. If it should be of a different value, please include it to the rule.
Piles (VA) - IBE_BOQReference	Checks if the parameter IBE_BOQReference is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_BOQReference is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter should follow the format: "XX.XX.XX". If it should be of a different value, please include it to the rule.
PileCaps (MOD) - Hosted not WorkPlane-Based	Checks if the pile caps in the model are hosted. This is not desirable for Structural Foundations as the elements will not be associate to a Level.	There are pile cap families that are hosted, and they will not report a level. This is not desirable. Please Review.

PileCaps (MOD) - Level	Checks if the pile caps in the model are assigned to the correct level.	There are pile caps associated to a wrong level or not associated to any level. The correct levels are: K01 or K02. Please Review.
PileCaps (MOD) - Elevation at Bottom	Checks if there are PileCaps with wrong elevation at bottom.	There are PileCaps that go deeper than designed. Please Review
PileCaps (MOD) - Elevation at Top	Checks if there are PileCaps with wrong elevation at top.	There are PileCaps with elevation at top above the ground level. Please Review
PileCaps (MOD) - Structural Workset	Checks if there are PileCaps outside the Structural Worksets	There are PileCaps that are not in the correct workset
PileCaps (CON) - Naming Convention	Checks if the PileCaps naming follow the naming convention. This check reports instances.	There are pile instances that do not follow the naming convention. Please check that the following naming convention is followed: Height(3or4digits)xWidth(3or4digits)-Bet_C30/37. Example: 400x1200-Bet_C30/37
PileCaps (CON) - PileCaps outside of defined checks	Checks if there are new pile cap types that are not included in the following checks: PileCaps (CON) - TypeName and Height, PileCaps (CON) - TypeName and Width. This check reports instances.	There are new pile cap types that cannot be checked in the next checks. The height dimension in the type name does not match 400, or the width dimension on the type name does not match 1200. Please review the elements and, if correct, add them to this rule and the following rules: PileCaps (CON) - TypeName and Height, PileCaps (CON) - TypeName and Width.
PileCaps (CON) - TypeName and Height	Checks if the PileCaps height matches the one described in the TypeName or there are new pile types that do not fit the check rule and cannot be validated. This check reports instances.	There are PileCaps where the height and the TypeName are not consistent, or there are new PileCaps types that can`t be validated with this check, because their TypeName height is different than 400, or their height is different than 0,4m. Please, check the reported PileCaps and, if they are correct, add them to the rule.
PileCaps (CON) - TypeName and Width	Checks if the PileCaps width matches the one described in the TypeName or there are new pile cap types that do not fit the check rule and cannot be validated. This check reports instances.	There are PileCaps where the width and the TypeName are not consistent, or there are new PileCaps types that can`t be validated with this check, because their TypeName width is different than 1200, or their width is different than 1,2m. Please, check the reported PileCaps and, if they are correct, add them to the rule.
PileCaps (VA) - IBE_ElementType	Checks if the parameter IBE_ElementType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ElementType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "Pilotna greda - pravokotna".
PileCaps (VA) - OmniClass Number	Checks if the built-in parameter "OmniClass Number" has a correct value. This check reports instances.	The built-in parameter "OmniClass Number" does not have a correct value. The value should be "23.25.05.17.14"
PileCaps (VA) - OmniClass Title	Checks if the built-in parameter "OmniClass Title" has a correct value. This check reports instances.	The built-in parameter "OmniClass Title" does not have a correct value. The value should be "Foundation Piles"
PileCaps (VA) - Mark	Checks if the built-in parameter "Mark" has a correct value. This check reports instances.	The built-in parameter "Mark" does not have a correct value. The value should follow the format "HBM-K01-001".
PileCaps (VA) - Description	Checks if the built-in parameter "Description" has a correct value. This check reports instances.	The built-in parameter "Description" does not have a correct value. The value should follow the format "Pilotna greda B/H=120x40 cm". This check only validates the values B=120cm and H=40cm. Please review the elements and, if correct, add them to this rule.
PileCaps (VA) - Type Comments	Checks if the built-in parameter "Type Comments" has a correct value. This check reports instances.	The built-in parameter "Type Comments" does not have a correct value. The value should follow the format "HBM-Tip A". Please review the elements and, if correct, add them to this rule.
PileCaps (VA) - Type Mark	Checks if the built-in parameter "Type Mark" has a correct value. This check reports instances.	The built-in parameter "Type Mark" does not have a correct value. The value should follow the format "HBM-01". Please review the elements and, if correct, add them to this rule.
PileCaps (VA) - IfcExportAs	Checks if the parameter IfcExportAs is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportAs is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "IfcFooting".
PileCaps (VA) - IfcExportType	Checks if the parameter IfcExportType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "FOOTING_BEAM".
PileCaps (VA) - IBE_ConcreteGrade	Checks if the parameter IBE_ConcreteGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteGrade is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "C30/37". If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_ExposureClass	Checks if the parameter IBE_ExposureClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExposureClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "XC0, XC2 or XC4". If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_ChlorideClass	Checks if the parameter IBE_ChlorideClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ChlorideClass is not defined for this category, or the value is wrong for the following elements. Accepted values for this parameter are: "Cl 0.00" up to "Cl 1.00" (this check also accepts 1 decimal number as valid: 0..2". If it should be of a different value, please include it to the rule.

PileCaps (VA) - IBE_ConcreteDmax	Checks if the parameter IBE_ConcreteDmax is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteDmax is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: D8, D16 or D32. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_ConcreteSlumpClass	Checks if the parameter IBE_ConcreteSlumpClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteSlumpClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: S1, S2, S3, S4 or S5.
PileCaps (VA) - IBE_ReinforcementGrade	Checks if the parameter IBE_ReinforcementGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementGrade is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: B500B. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_ReinforcementRatio	Checks if the parameter IBE_ReinforcementRatio is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementRatio is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0-300 kg/m3. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_RebarCover	Checks if the parameter IBE_RebarCover is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarCover is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0,04m. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_ConstructionProcess	Checks if the parameter IBE_ConstructionProcess is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConstructionProcess is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: InSitu. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_ExecutionClass	Checks if the parameter IBE_ExecutionClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExecutionClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: EXC2. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_SurfaceTreatment	Checks if the parameter IBE_SurfaceTreatment is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SurfaceTreatment is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: VB0. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_CuringClass	Checks if the parameter IBE_CuringClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_CuringClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_ToleranceClass	Checks if the parameter IIBE_ToleranceClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ToleranceClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_SideFormworkType	Checks if the parameter IBE_SideFormworkType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SideFormworkType is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "Opaž roba pilotne grede". If it should be of a different value, please include it to the rule.
PileCaps (VA) - Keynote	Checks if the built-in parameter "Keynote" has a correct value. This check reports instances.	The built-in parameter "Keynote" does not have a correct value. The value should follow the format "01.07.01.XXX".
PileCaps (VA) - IBE_OpisPostavkeIZS	Checks if the parameter IBE_OpisPostavkeIZS is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_OpisPostavkeIZS is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "Pilotna greda B/H=120x40 cm, C30/37". If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_RebarQuantityEstimate	Checks if the parameter IBE_RebarQuantityEstimate is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarQuantityEstimate is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: >=0. If it should be of a different value, please include it to the rule.
PileCaps (VA) - IBE_BOQReference	Checks if the parameter IBE_BOQReference is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_BOQReference is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter should follow the format: "XX.XX.XX". If it should be of a different value, please include it to the rule.
GroundBeams (MOD) - Hosted not WorkPlane-Based	Checks if the ground beams in the model are hosted. This is not desirable for Structural Foundations as the elements will not be associate to a Level.	There are pile cap families that are hosted, and they will not report a level. This is not desirable. Please Review.
GroundBeams (MOD) - Level	Checks if the ground beams in the model are assigned to the correct level.	There are ground beams associated to a wrong level or are not associated to it. The correct levels are: K01. Please Review.
GroundBeams (MOD) - Elevation at Bottom	Checks if there are ground beams with wrong elevation at bottom.	There are ground beams that go deeper than designed. Please Review
GroundBeams (MOD) - Elevation at Top	Checks if there are ground beams with wrong elevation at top.	There are ground beams with elevation at top above the ground level. Please Review
GroundBeams (MOD) - Structural Workset	Checks if there are GroundBeams outside the Structural Worksets	There are GroundBeams that are not in the correct workset

GroundBeams (CON) - Naming Convention	Checks if the ground beams naming follow the naming convention. This check reports instances.	There are ground beam instances that do not follow the naming convention. Please check that the following naming convention is followed: Width(3or4digits)xHeight(3or4digits)-Bet_C30/37. Example: 1900x1200-Bet_C30/37
GroundBeams (CON) - GroundBeams outside of defined checks	Checks if there are new ground beam types that are not included in the following checks: GroundBeams (CON) - TypeName and Width, GroundBeams (CON) - TypeName and Height. This check reports instances.	There are new ground beam types that cannot be checked in the next checks. The Width dimension in the type name does not match 400, 1000, 1200, 1400, 1500 or 1900 or the Height dimension on the type name does not match 500, 1000 or 1200. Please review the elements and, if correct, add them to this rule and the following rules: GroundBeams (CON) - TypeName and Height, GroundBeams (CON) - TypeName and Width.
GroundBeams (CON) - TypeName and Width	Checks if the ground beams width matches the one described in the TypeName or there are new ground beam types that do not fit the check rule and cannot be validated. This check reports instances.	There are ground beams where the width and the TypeName are not consistent, or there are new GroundBeams types that can`t be validated with this check, because their TypeName width is different than 400, 1000, 1200, 1400, 1500 or 1900, or their height is different than 0,4, 1,0, 1,2, 1,4, 1,5, or 1,9 m. Please, check the reported GroundBeams and, if they are correct, add them to the rule.
GroundBeams (CON) - TypeName and Height	Checks if the ground beams height matches the one described in the TypeName or there are new ground beam types that do not fit the check rule and cannot be validated. This check reports instances.	There are ground beams where the hieght and the TypeName are not consistent, or there are new GroundBeams types that can`t be validated with this check, because their TypeName height is different than 500, 1000, 1200, or their height is different than 0,5, 1,0 or 1,2 m. Please, check the reported GroundBeams and, if they are correct, add them to the rule.
GroundBeams (VA) - IBE_ElementType	Checks if the parameter IBE_ElementType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ElementType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "Pasovni temelj".
GroundBeams (VA) - OmniClass Number	Checks if the built-in parameter "OmniClass Number" has a correct value. This check reports instances.	The built-in parameter "OmniClass Number" does not have a correct value. The value should be "23.25.05.17.14"
GroundBeams (VA) - OmniClass Title	Checks if the built-in parameter "OmniClass Title" has a correct value. This check reports instances.	The built-in parameter "OmniClass Title" does not have a correct value. The value should be "Foundation Piles"
GroundBeams (VA) - Mark	Checks if the built-in parameter "Mark" has a correct value. This check reports instances.	The built-in parameter "Mark" does not have a correct value. The value should follow the format "STF-K01-001".
GroundBeams (VA) - Description	Checks if the built-in parameter "Description" has a correct value. This check reports instances.	The built-in parameter "Description" does not have a correct value. The value should follow the format "Pasovni temelj B/H=100/50 cm". This check only validates the values B=50, 100, 120, 140, 150 and 190 cm and H=50, 100, 120 cm. Please review the elements and, if correct, add them to this rule.
GroundBeams (VA) - Type Comments	Checks if the built-in parameter "Type Comments" has a correct value. This check reports instances.	The built-in parameter "Type Comments" does not have a correct value. The value should follow the format "STF-Tip 1". Please review the elements and, if correct, add them to this rule.
GroundBeams (VA) - Type Mark	Checks if the built-in parameter "Type Mark" has a correct value. This check reports instances.	The built-in parameter "Type Mark" does not have a correct value. The value should follow the format "STF-01". Please review the elements and, if correct, add them to this rule.
GroundBeams (VA) - IfcExportAs	Checks if the parameter IfcExportAs is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportAs is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "IfcFooting".
GroundBeams (VA) - IfcExportType	Checks if the parameter IfcExportType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "STRIP_FOOTING".
GroundBeams (VA) - IBE_ConcreteGrade	Checks if the parameter IBE_ConcreteGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteGrade is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "C30/37". If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_ExposureClass	Checks if the parameter IBE_ExposureClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExposureClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "XC0, XC2 or XC4". If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_ChlorideClass	Checks if the parameter IBE_ChlorideClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ChlorideClass is not defined for this category, or the value is wrong for the following elements. Accepted values for this parameter are: "Cl 0.00" up to "Cl 1.00" (this check also accepts 1 decimal number as valid: 0..2". If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_ConcreteDmax	Checks if the parameter IBE_ConcreteDmax is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteDmax is not defined for this category, or the value is wrong for the following elements. The ccepted values for this parameter are: D8, D16 or D32. If it should be of a different value, please include it to the rule.

GroundBeams (VA) - IBE_ConcreteSlumpClass	Checks if the parameter IBE_ConcreteSlumpClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteSlumpClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: S1, S2, S3, S4 or S5.
GroundBeams (VA) - IBE_ReinforcementGrade	Checks if the parameter IBE_ReinforcementGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementGrade is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: B500B. If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_ReinforcementRatio	Checks if the parameter IBE_ReinforcementRatio is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementRatio is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0-300 kg/m3. If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_RebarCover	Checks if the parameter IBE_RebarCover is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarCover is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0,04m. If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_ConstructionProcess	Checks if the parameter IBE_ConstructionProcess is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConstructionProcess is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: InSitu. If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_ExecutionClass	Checks if the parameter IBE_ExecutionClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExecutionClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: EXC2. If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_SurfaceTreatment	Checks if the parameter IBE_SurfaceTreatment is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SurfaceTreatment is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: VB0. If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_CuringClass	Checks if the parameter IBE_CuringClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_CuringClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_ToleranceClass	Checks if the parameter IIBE_ToleranceClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ToleranceClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_SideFormworkType	Checks if the parameter IBE_SideFormworkType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SideFormworkType is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "Opaž pasovnih temeljev". If it should be of a different value, please include it to the rule.
GroundBeams (VA) - Keynote	Checks if the built-in parameter "Keynote" has a correct value. This check reports instances.	The built-in parameter "Keynote" does not have a correct value. The value should follow the format "01.07.01.05G".
GroundBeams (VA) - IBE_OpisPostavkeIZS	Checks if the parameter IBE_OpisPostavkeIZS is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_OpisPostavkeIZS is not defined for this category, or the value is wrong for the following elements. The value format for this parameter is: "Pasovni temelj B/H=100x50 cm, C30/37". If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_RebarQuantityEstimate	Checks if the parameter IBE_RebarQuantityEstimate is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarQuantityEstimate is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: >=0. If it should be of a different value, please include it to the rule.
GroundBeams (VA) - IBE_BOQReference	Checks if the parameter IBE_BOQReference is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_BOQReference is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter should follow the format: "XX.XX.XX". If it should be of a different value, please include it to the rule.
PocketFoundations (MOD) - Hosted not WorkPlane-Based	Checks if the pocket foundations in the model are hosted. This is not desirable for Structural Foundations as the elements will not be associate to a Level.	There are pile cap families that are hosted, and they will not report a level. This is not desirable. Please Review.
PocketFoundations (MOD) - Level	Checks if the pocket foundations in the model are assigned to the correct level.	There are pocket foundations associated to a wrong level or are not associated to it. The correct levels are: P00. Please Review.
PocketFoundations (MOD) - Elevation at Bottom	Checks if there are pocket foundations with wrong elevation at bottom.	There are pocket foundations that go deeper than designed. Please Review
PocketFoundations (MOD) - Elevation at Top	Checks if there are pocket foundations with wrong elevation at top.	There are pocket foundations with elevation at top above the ground level. Please Review
PocketFoundations (MOD) - Structural Workset	Checks if there are PocketFoundations outside the Structural Worksets	There are PocketFoundations that are not in the correct workset
PocketFoundations (CON) - Naming Convention	Checks if the pocket foundations naming follow the naming convention. This check reports instances.	There are pocket foundation instances that do not follow the naming convention. Please check that the following naming convention is followed: Width(3or4digits)xLength(3or4digits)xThickness(3or4digits). Example: 1700x1700x2200

PocketFoundations (CON) - PocketFoundations outside of defined checks	Checks if there are new pocket foundation types that are not included in the following checks: PocketFoundations (CON) - TypeName and Width, PocketFoundations (CON) - TypeName and Length or PocketFoundations (CON) - TypeName and Thickness. This check reports instances.	There are new PocketFoundations types that cannot be checked in the next checks. The dimensions in the type name does not match 1700x1700x2200, 1700x1700x2500, 1800x1800x2500, 1900x1900x2700 or 1900x1900x2900. Please review the elements and, if correct, add them to this rule and the following rules: PocketFoundations (CON) - TypeName and Width, PocketFoundations (CON) - TypeName and Width or PocketFoundations (CON) - TypeName and Thickness.
PocketFoundations (CON) - TypeName and Length	Checks if the pocket foundations length matches the one described in the TypeName or there are new pocket foundation types that do not fit the check rule and cannot be validated. This check reports instances.	There are pocket foundations where the length and the TypeName are not consistent, or there are new pocket foundation types that can`t be validated with this check, because their TypeName length is different than 1700, 1800 or 1900, or their height is different than 1,7, 1,8, or 1,9 m. Please, check the reported pocket foundations and, if they are correct, add them to the rule.
PocketFoundations (CON) - TypeName and Width	Checks if the pocket foundations width matches the one described in the TypeName or there are new pocket foundation types that do not fit the check rule and cannot be validated. This check reports instances.	There are pocket foundations where the width and the TypeName are not consistent, or there are new pocket foundation types that can`t be validated with this check, because their TypeName width is different than 1700, 1800, 1900, or their height is different than 1,7, 1,8 or 1,9 m. Please, check the reported pocket foundations and, if they are correct, add them to the rule.
PocketFoundations (CON) - TypeName and Thickness	Checks if the pocket foundations thickness matches the one described in the TypeName or there are new pocket foundation types that do not fit the check rule and cannot be validated. This check reports instances.	There are pocket foundations where the thickness and the TypeName are not consistent, or there are new pocket foundation types that can`t be validated with this check, because their TypeName thickness is different than 2200, 2500, 2700 or 2900, or their thickness is different than 2,2, 2,5, 2,7 or 2,9 m. Please, check the reported pocket foundations and, if they are correct, add them to the rule.
PocketFoundations (VA) - IBE_ElementType	Checks if the parameter IBE_ElementType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ElementType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "Pasovni temelj".
PocketFoundations (VA) - OmniClass Number	Checks if the built-in parameter "OmniClass Number" has a correct value. This check reports instances.	The built-in parameter "OmniClass Number" does not have a correct value. The value should be "23.25.05.17"
PocketFoundations (VA) - OmniClass Title	Checks if the built-in parameter "OmniClass Title" has a correct value. This check reports instances.	The built-in parameter "OmniClass Title" does not have a correct value. The value should be "Foundation Piles"
PocketFoundations (VA) - Mark	Checks if the built-in parameter "Mark" has a correct value. This check reports instances.	The built-in parameter "Mark" does not have a correct value. The value should follow the format "PCF-P00-001".
PocketFoundations (VA) - Description	Checks if the built-in parameter "Description" has a correct value. This check reports instances.	The built-in parameter "Description" does not have a correct value. The value should follow the format "Temeljna casa L/B/H=180/180/250 cm". This check only validates the values L=170,180 and 190, B=170,180 and 190, and H=220, 250, 270 and 290 cm. Please review the elements and, if correct, add them to this rule.
PocketFoundations (VA) - Type Comments	Checks if the built-in parameter "Type Comments" has a correct value. This check reports instances.	The built-in parameter "Type Comments" does not have a correct value. The value should follow the format "PCF-Tip 1". Please review the elements and, if correct, add them to this rule.
PocketFoundations (VA) - Type Mark	Checks if the built-in parameter "Type Mark" has a correct value. This check reports instances.	The built-in parameter "Type Mark" does not have a correct value. The value should follow the format "PCF-01". Please review the elements and, if correct, add them to this rule.
PocketFoundations (VA) - IfcExportAs	Checks if the parameter IfcExportAs is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportAs is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "IfcFooting".
PocketFoundations (VA) - IfcExportType	Checks if the parameter IfcExportType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "PAD_FOOTING".
PocketFoundations (VA) - IBE_ConcreteGrade	Checks if the parameter IBE_ConcreteGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteGrade is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "C30/37". If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_ExposureClass	Checks if the parameter IBE_ExposureClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExposureClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "XC0, XC2 or XC4". If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_ChlorideClass	Checks if the parameter IBE_ChlorideClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ChlorideClass is not defined for this category, or the value is wrong for the following elements. Accepted values for this parameter are: "Cl 0.00" up to "Cl 1.00" (this check also accepts 1 decimal number as valid: 0..2". If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_ConcreteDmax	Checks if the parameter IBE_ConcreteDmax is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteDmax is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: D8, D16 or D32. If it should be of a different value, please include it to the rule.

PocketFoundations (VA) - IBE_ConcreteSlumpClass	Checks if the parameter IBE_ConcreteSlumpClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteSlumpClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: S1, S2, S3, S4 or S5.
PocketFoundations (VA) - IBE_ReinforcementGrade	Checks if the parameter IBE_ReinforcementGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementGrade is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: B500B. If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_ReinforcementRatio	Checks if the parameter IBE_ReinforcementRatio is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementRatio is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0-300 kg/m3. If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_RebarCover	Checks if the parameter IBE_RebarCover is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarCover is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0,04m. If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_ConstructionProcess	Checks if the parameter IBE_ConstructionProcess is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConstructionProcess is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: InSitu. If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_ExecutionClass	Checks if the parameter IBE_ExecutionClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExecutionClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: EXC2. If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_SurfaceTreatment	Checks if the parameter IBE_SurfaceTreatment is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SurfaceTreatment is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: VB0. If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_CuringClass	Checks if the parameter IBE_CuringClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_CuringClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_ToleranceClass	Checks if the parameter IIBE_ToleranceClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ToleranceClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_SideFormworkType	Checks if the parameter IBE_SideFormworkType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SideFormworkType is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "Opaž temeljne case". If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - Keynote	Checks if the built-in parameter "Keynote" has a correct value. This check reports instances.	The built-in parameter "Keynote" does not have a correct value. The value should follow the format "01.07.01.05X".
PocketFoundations (VA) - IBE_OpisPostavkeIZS	Checks if the parameter IBE_OpisPostavkeIZS is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_OpisPostavkeIZS is not defined for this category, or the value is wrong for the following elements. The value format for this parameter is: "Temeljna casa L/B/H=180/180/250 cm, C30/37". If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_RebarQuantityEstimate	Checks if the parameter IBE_RebarQuantityEstimate is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarQuantityEstimate is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: >=0. If it should be of a different value, please include it to the rule.
PocketFoundations (VA) - IBE_BOQReference	Checks if the parameter IBE_BOQReference is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_BOQReference is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter should follow the format: "XX.XX.XX". If it should be of a different value, please include it to the rule.
FoundationSlabs (MOD) - Level	Checks if the foundation slabs in the model are assigned to the correct level.	There are foundation slabs associated to a wrong level or are not associated to it. The correct levels are: K01, K02 or P00. Please Review.
FoundationSlabs (MOD) - Elevation at Bottom	Checks if there are foundation slabs with wrong elevation at bottom.	There are foundation slabs that go deeper than designed. Please Review
FoundationSlabs (MOD) - Elevation at Top	Checks if there are foundation slabs with wrong elevation at top.	There are foundation slabs with elevation at top above the ground level. Please Review
FoundationSlabs (MOD) - Structural Workset	Checks if there are FoundationSlabs outside the Structural Worksets	There are FoundationSlabs that are not in the correct workset
FoundationSlabs (CON) - Naming Convention	Checks if the foundation slabs naming follow the naming convention. This check reports instances.	There are foundation slab instances that do not follow the naming convention. Please check that the following naming convention is followed: SFO-Thickness(3or4digits)-Bet_C30/37. Example: SFO-300-Bet_C30/37.

FoundationSlabs (CON) - FoundationSlabs outside of defined checks	Checks if there are new foundation slab types that are not included in the following checks: FoundationSlabs (CON) - TypeName and Thickness. This check reports instances.	There are new foundation slab types that cannot be checked in the next checks. The thickness dimension in the type name does not match 300, 400, 480, 600, 1000, 1080. Please review the elements and, if correct, add them to this rule and the following rules: FoundationSlabs (CON) - TypeName and Thickness.
FoundationSlabs (CON) - TypeName and Thickness	Checks if the foundation slabs thickness matches the one described in the TypeName or there are new pocket foundation types that do not fit the check rule and cannot be validated. This check reports instances.	There are foundation slabs where the thickness and the TypeName are not consistent, or there are new foundation slab types that can`t be validated with this check, because their TypeName thickness is different than 300, 400, 480, 600, 1000, 1080, or their thickness is different than 0,3, 0,4, 0,48, 0,6, 1,0 or 1,08 m. Please, check the reported foundation slabs and, if they are correct, add them to the rule.
FoundationSlabs (VA) - IBE_ElementType	Checks if the parameter IBE_ElementType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ElementType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "Temeljna plošča".
FoundationSlabs (VA) - Mark	Checks if the built-in parameter "Mark" has a correct value. This check reports instances.	The built-in parameter "Mark" does not have a correct value. The value should follow the format "FSL-P00-001". The accepted levels are K02, K01 and P00.
FoundationSlabs (VA) - Description	Checks if the built-in parameter "Description" has a correct value. This check reports instances.	The built-in parameter "Description" does not have a correct value. The value should follow the format "Temeljna plosca t=80 cm". This check only validates the values t=30, 40, 48, 60, 100 and 108 cm. Please review the elements and, if correct, add them to this rule.
FoundationSlabs (VA) - Type Comments	Checks if the built-in parameter "Type Comments" has a correct value. This check reports instances.	The built-in parameter "Type Comments" does not have a correct value. The value should follow the format "FSL-T30". Please review the elements and, if correct, add them to this rule.
FoundationSlabs (VA) - Type Mark	Checks if the built-in parameter "Type Mark" has a correct value. This check reports instances.	The built-in parameter "Type Mark" does not have a correct value. The value should follow the format "FSL-01". Please review the elements and, if correct, add them to this rule.
FoundationSlabs (VA) - IfcExportAs	Checks if the parameter IfcExportAs is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportAs is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "IfcSlab".
FoundationSlabs (VA) - IfcExportType	Checks if the parameter IfcExportType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IfcExportType is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "BASESLAB".
FoundationSlabs (VA) - IBE_ConcreteGrade	Checks if the parameter IBE_ConcreteGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteGrade is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "C30/37". If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_ExposureClass	Checks if the parameter IBE_ExposureClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExposureClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "XC0, XC2 or XC4". If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_ChlorideClass	Checks if the parameter IBE_ChlorideClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ChlorideClass is not defined for this category, or the value is wrong for the following elements. Accepted values for this parameter are: "Cl 0.00" up to "Cl 1.00" (this check also accepts 1 decimal number as valid: 0..2". If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_ConcreteDmax	Checks if the parameter IBE_ConcreteDmax is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteDmax is not defined for this category, or the value is wrong for the following elements. The ccepted values for this parameter are: D8, D16 or D32. If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_ConcreteSlumpClass	Checks if the parameter IBE_ConcreteSlumpClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteSlumpClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: S1, S2, S3, S4 or S5.
FoundationSlabs (VA) - IBE_ConcreteGrade	Checks if the parameter IBE_ConcreteGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConcreteGrade is not defined for this category, or the value is wrong for the following elements. The value for this parameter should be "C30/37". If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - Structural	Checks if the value of the "built-in" boolean parameter Structural is correct. This check reports instances.	The value of the "built-in" parameter Structural is not correct. The box should be ticked. If it should be of a different value, please include it to the rule as an exception.
FoundationSlabs (VA) - IBE_ReinforcementGrade	Checks if the parameter IBE_ReinforcementGrade is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementGrade is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: B500B. If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_ReinforcementRatio	Checks if the parameter IBE_ReinforcementRatio is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ReinforcementRatio is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0-300 kg/m3. If it should be of a different value, please include it to the rule.

FoundationSlabs (VA) - IBE_RebarCover	Checks if the parameter IBE_RebarCover is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarCover is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 0,04m. If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_ConstructionProcess	Checks if the parameter IBE_ConstructionProcess is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ConstructionProcess is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: InSitu. If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_ExecutionClass	Checks if the parameter IBE_ExecutionClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ExecutionClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: EXC2. If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_SurfaceTreatment	Checks if the parameter IBE_SurfaceTreatment is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SurfaceTreatment is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: VB0. If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_CuringClass	Checks if the parameter IBE_CuringClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_CuringClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_ToleranceClass	Checks if the parameter IIBE_ToleranceClass is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_ToleranceClass is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: 1. If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_SideFormworkType	Checks if the parameter IBE_SideFormworkType is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_SideFormworkType is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: "Opaz roba temeljne plosce". If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - Keynote	Checks if the built-in parameter "Keynote" has a correct value. This check reports instances.	The built-in parameter "Keynote" does not have a correct value. The value should follow the format "01.07.01.07X".
FoundationSlabs (VA) - IBE_OpisPostavkeIZS	Checks if the parameter IBE_OpisPostavkeIZS is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_OpisPostavkeIZS is not defined for this category, or the value is wrong for the following elements. The value format for this parameter is: "Temeljna plosca t=30 cm, C30/37". If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_RebarQuantityEstimate	Checks if the parameter IBE_RebarQuantityEstimate is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_RebarQuantityEstimate is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter are: >=0. If it should be of a different value, please include it to the rule.
FoundationSlabs (VA) - IBE_BOQReference	Checks if the parameter IBE_BOQReference is defined and if it is, if the value of the parameter is correct. This check reports instances.	The parameter IBE_BOQReference is not defined for this category, or the value is wrong for the following elements. The accepted values for this parameter should follow the format: "XX.XX.XX". If it should be of a different value, please include it to the rule.